

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CISCO SYSTEMS, INC., and
CISCO TECHNOLOGY, INC.,

Plaintiffs,

V.

TELCORDIA TECHNOLOGIES, INC.,

Defendant.

Civil Action No.

DEMAND FOR JURY TRIAL

COMPLAINT

Plaintiffs Cisco Systems, Inc. ("Cisco Systems") and Cisco Technology, Inc. ("Cisco Technology") (collectively "Cisco"), by and for their complaint for infringement of United States Patent Nos. 5,142,622 ("the '622 Patent") and 6,377,988 ("the '988 Patent") against defendant Telcordia Technologies, Inc. ("Telcordia"), allege as follows:

THE PARTIES

1. Cisco Systems is a corporation organized under the laws of California. Cisco Systems maintains its headquarters at 170 West Tasman Drive, San Jose, California 95134.

2. Cisco Technology is a corporation organized under the laws of California. Cisco Technology maintains its headquarters at 170 West Tasman Drive, San Jose, California 95134. Cisco Technology is a wholly-owned subsidiary of Cisco Systems.

3. Upon information and belief, Telcordia is a corporation organized under the laws of Delaware, with its principal place of business at One Telcordia Drive, Piscataway, New Jersey 08854.

JURISDICTION AND VENUE

4. This Court has subject matter jurisdiction under 28 U.S.C. §§ 1331 and 1338(a).

5. This Court has personal jurisdiction over Telcordia based on its significant contacts with this district, including, but not limited to, Telcordia's incorporation in this district, Telcordia's distribution, sale, and offer for sale of infringing communication network products within this district, and Telcordia's active inducement of and/or contribution to infringement within this district.

6. Venue is proper in this judicial district under 28 U.S.C. §§ 1391 and 1400(b).

CAUSE OF ACTION FOR PATENT INFRINGEMENT

7. Cisco repeats and re-alleges the allegations of Paragraphs 1-6 as though fully set forth herein.

8. The '622 Patent, entitled "System for Interconnecting Applications Across Different Networks of Data Processing Systems by Mapping Protocols Across Different Network Domains" was duly and lawfully issued to inventor Gary L. Owens by the United States Patent and Trademark Office ("PTO") on August 25, 1992. Cisco Systems owns the '622 Patent and holds all right, title, and interest in the '622 Patent, including the rights to sue for infringement and seek damages and injunctive relief. A copy of the '622 Patent is attached hereto as Exhibit A.

9. The '988 Patent, entitled "Controlling Elements of a Telecommunications Network" was duly and lawfully issued to inventors Adam Spector and Paul Abraham by the PTO on April 23, 2002. Cisco Technology owns the '988 Patent and holds all right, title, and interest in the '988 Patent, including the rights to sue for infringement and seek damages and injunctive relief. A copy of the '988 Patent is attached hereto as Exhibit B.

FIRST CLAIM

(Patent Infringement Of The '622 Patent)

10. On information and belief, Telcordia has been and is infringing one or more claims of the '622 Patent, directly and/or indirectly, pursuant to 35 U.S.C. § 271, in connection with certain of its communication network products, including, without limitation, Telcordia Element Communicator CE, Granite Inventory, and NMA System CE.

11. On information and belief, Telcordia's infringement of the '622 Patent has been and is willful, and will continue unless enjoined by this Court. Cisco has suffered, and will continue to suffer, irreparable injury as a result of this willful infringement. Pursuant to 35 U.S.C. § 284, Cisco is entitled to damages for infringement and treble damages. Pursuant to 35 U.S.C. § 283, Cisco is entitled to a permanent injunction against further infringement.

12. This case is exceptional and, therefore, Cisco is entitled to attorneys' fees pursuant to 35 U.S.C. § 285.

SECOND CLAIM

(Patent Infringement Of The '988 Patent)

13. On information and belief, Telcordia has been and is infringing one or more claims of the '988 Patent, directly and/or indirectly, pursuant to 35 U.S.C. § 271, in connection with certain of its communication network products, including, without limitation, Telcordia Element Communicator CE, Granite Inventory, and NMA System CE.

14. On information and belief, Telcordia's infringement of the '988 Patent has been and is willful, and will continue unless enjoined by this Court. Cisco has suffered, and will continue to suffer, irreparable injury as a result of this willful infringement. Pursuant to 35 U.S.C. § 284, Cisco is entitled to damages for infringement and treble damages. Pursuant to 35 U.S.C. § 283, Cisco is entitled to a permanent injunction against further infringement.

15. This case is exceptional and, therefore, Cisco is entitled to attorneys' fees pursuant to 35 U.S.C. § 285.

PRAYER FOR RELIEF

WHEREFORE, Cisco prays for relief as follows:

- A. That Telcordia be adjudged to have infringed the '622 and '988 Patents;
- B. That Telcordia, its officers, agents, servants, employees, attorneys, and those persons in active concert or participation with any of them, be preliminarily and permanently restrained and enjoined from directly or indirectly infringing the '622 and '988 Patents;
- C. An award of damages to compensate Cisco for Telcordia's infringement, pursuant to 35 U.S.C. § 284, said damages to be trebled because of Telcordia's willful infringement;
- D. An assessment of pre-judgment and post-judgment interest and costs against Telcordia, together with an award of such interest and costs;
- E. That Telcordia be directed to pay Cisco's attorneys' fees incurred in connection with this lawsuit pursuant to 35 U.S.C. § 285; and
- F. That Cisco have such other and further relief as this Court may deem just and proper.

JURY DEMAND

Cisco demands a trial by jury on all issues amenable to trial by jury.

MORRIS, NICHOLS, ARSHT & TUNNELL LLP



Jack B. Blumenfeld (No. 1014)

Leslie A. Polizoti (No. 4299)

1201 North Market Street

P.O. Box 1347

Wilmington, DE 19899-1347

(302) 658-9200

Attorneys for Plaintiffs

Cisco Systems, Inc. and Cisco Technology, Inc.

Of Counsel:

Matthew D. Powers

Edward R. Reines

WEIL, GOTSHAL & MANGES

201 Redwood Shores Parkway

Redwood Shores, CA 94065

(650) 802-3000

Paul E. Torchia

WEIL, GOTSHAL & MANGES

767 Fifth Avenue

New York, NY 10153

(212) 310-8000

February 23, 2007

EXHIBIT A



US005142622A

United States Patent [19]

Owens

[11] Patent Number: **5,142,622**
 [45] Date of Patent: **Aug. 25, 1992**

[54] **SYSTEM FOR INTERCONNECTING APPLICATIONS ACROSS DIFFERENT NETWORKS OF DATA PROCESSING SYSTEMS BY MAPPING PROTOCOLS ACROSS DIFFERENT NETWORK DOMAINS**

[75] Inventor: **Gary L. Owens**, Mountain View, Calif.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **304,696**

[22] Filed: **Jan. 31, 1989**

[51] Int. Cl.⁵ **G06F 13/12**

[52] U.S. Cl. **395/200; 364/280.9; 364/284.3; 364/284.4; 364/284; 364/DIG. 1; 395/700; 395/500**

[58] Field of Search **395/200, 700, 500; 364/200 MS File, 900 MS File**

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,274,139	6/1981	Hodgkinson	364/200
4,322,792	3/1982	Baun	364/200
4,415,986	11/1983	Chadra	364/900
4,500,960	2/1985	Babecki	364/200
4,530,051	7/1985	Johnson	364/200
4,575,793	3/1986	Morel	364/200
4,586,134	4/1986	Norstedt	364/200
4,604,686	8/1986	Reiter	364/200
4,612,416	9/1986	Emerson	379/88
4,631,666	12/1986	Harris	364/200
4,677,588	6/1987	Benjamin et al.	364/900
4,679,189	7/1987	Olson	370/60
4,703,475	10/1987	Dretzka et al.	370/60
4,706,081	11/1987	Hart	340/825.03
4,736,369	4/1988	Barzilai	370/94.1
4,760,395	7/1988	Katzeff	370/60
4,768,150	8/1988	Chang et al.	364/300
4,790,003	12/1988	Kepley	379/88
4,811,216	3/1989	Bishop	364/200
4,825,354	4/1989	Agrawal	364/200
4,831,518	5/1989	Yu	364/200
4,849,877	7/1989	Bishop	364/200
4,855,906	8/1989	Burke	364/200
4,882,674	11/1989	Quint	364/900
4,893,307	1/1990	McKay	370/60
4,901,231	2/1990	Bishop	364/200

OTHER PUBLICATIONS

Introducing the UNIX System by H. McGilton and R.

Morgan pp. 1-7 McGraw-Hill Software Series.
 The UNIX Book by Mike Banahan and A. Rutter, J. Wiley & Sons Inc. 1983 pp. 107-120.
 The Design of the UNIX Operating System, by Bach, M. J., 1986, Prentice Hall, Englewood Cliffs, N.J.
 Dictionary of Computing, 8th Ed., Mar. 1987, IBM Document Composition Facility pp. 238, 326.
 Data Communications, vol. 16, No. 5, May 1987, New York, US, pp. 120-142, "SNA to OSI: IBM Building Upper-Layer Gateways" by Thomas J. Routt.
 The Sixth International Phoenix Conference on Computers and Communications Feb. 25, 1987, Scottsdale, Ariz., USA, pp. 354-360 by R. Martinez et al.
 IEEE Micro, vol. 5, No. 2, Apr. 1985, New York, US, pp. 53-66, "A Multimicrocomputer-based Structure for Computer Networking" by A. Faro et al.
 IEEE Infocom '87, Proceedings, Sixth Annual Conference, Mar. 31, 1987, San Francisco, Calif., USA, pp. 1045-1052, "Gateways for the OSI Transport Service" by G. V. Bochman et al.
 Proceedings, Ninth Data Communications Symposium, Sep. 10, 1985, Whistler Mountain British Columbia, pp. 2-8 "Development of a TCP/IP for the IBM/370" by R. K. Brandiff et al.

Primary Examiner—Thomas C. Lee

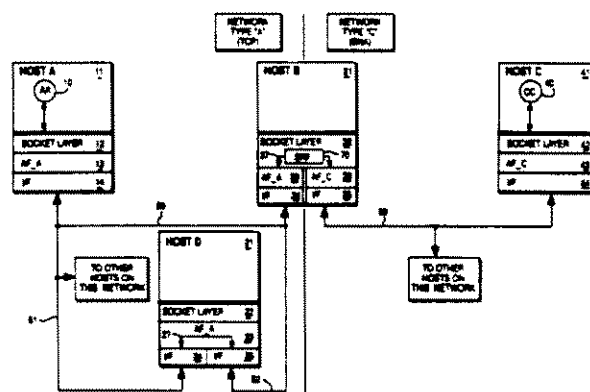
Assistant Examiner—Eric Coleman

Attorney, Agent, or Firm—Wayne P. Bailey; Marilyn D. Smith

[57] ABSTRACT

The system and method of this invention automatically routes a connection between data processing systems in different network domains. As an example, an application running on a data processing system utilizing a network domain such as TCP (Transmission Control Protocol), can automatically make a connection to another data processing system utilizing a different network domain such as SNA (Systems Network Architecture). The connection is automatically performed in the layer containing the communication end point objects. In a preferred embodiment, the connection is automatically performed in the socket layer of the AIX operating system, or in the socket layer of other operating systems based upon the Berkeley version of the UNIX operating system.

8 Claims, 7 Drawing Sheets



U.S. Patent

Aug. 25, 1992

Sheet 1 of 7

5,142,622

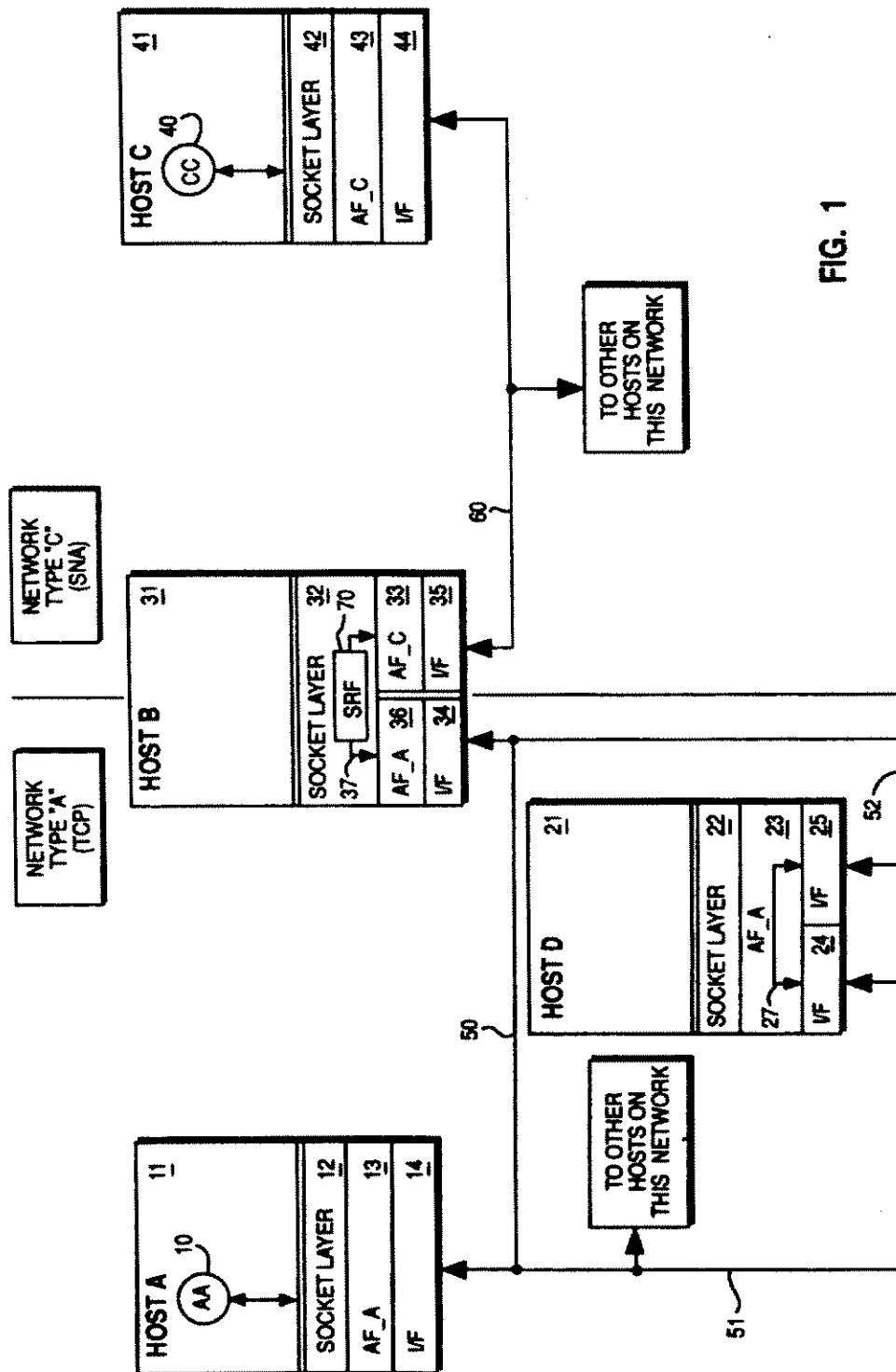
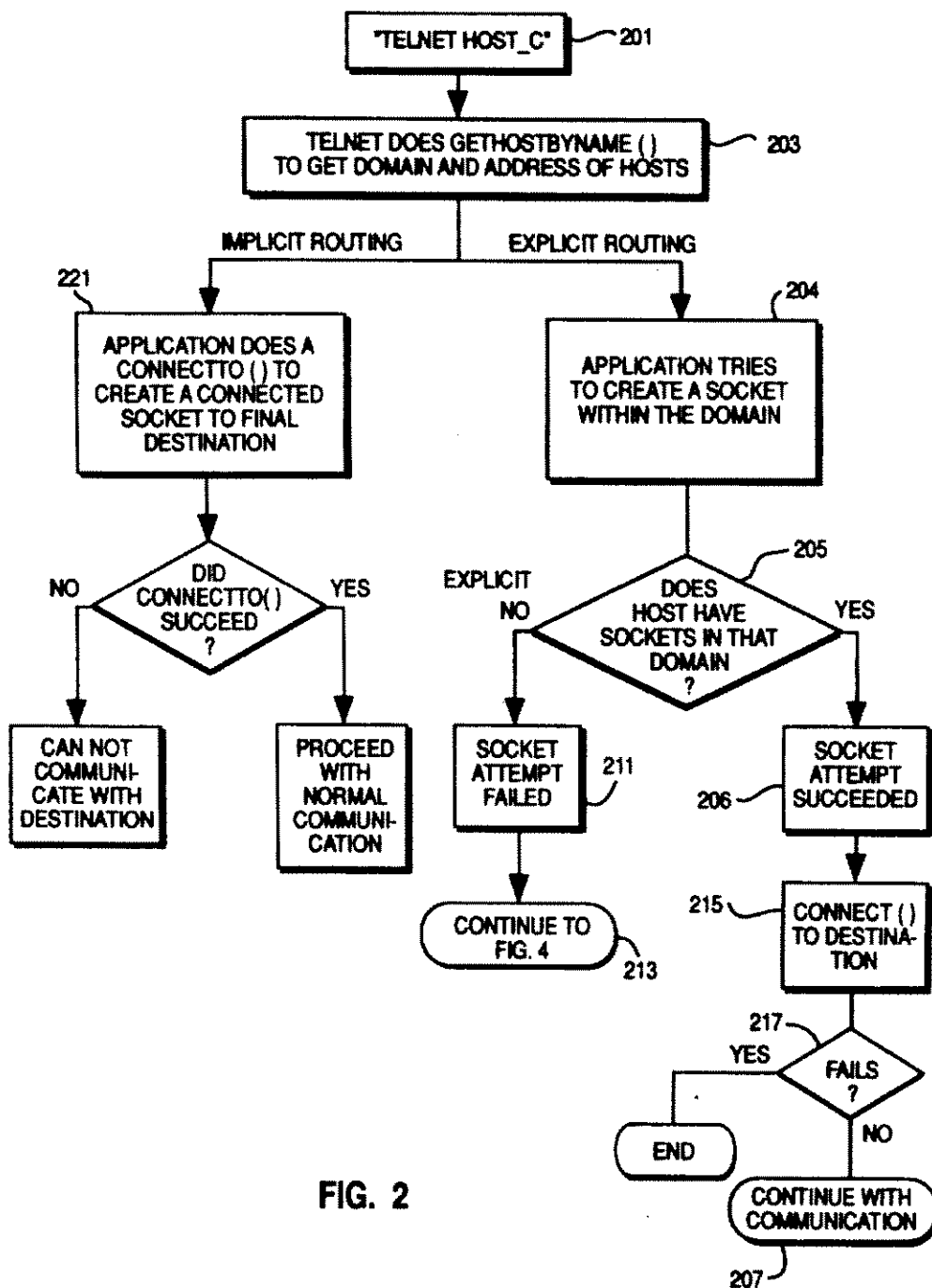
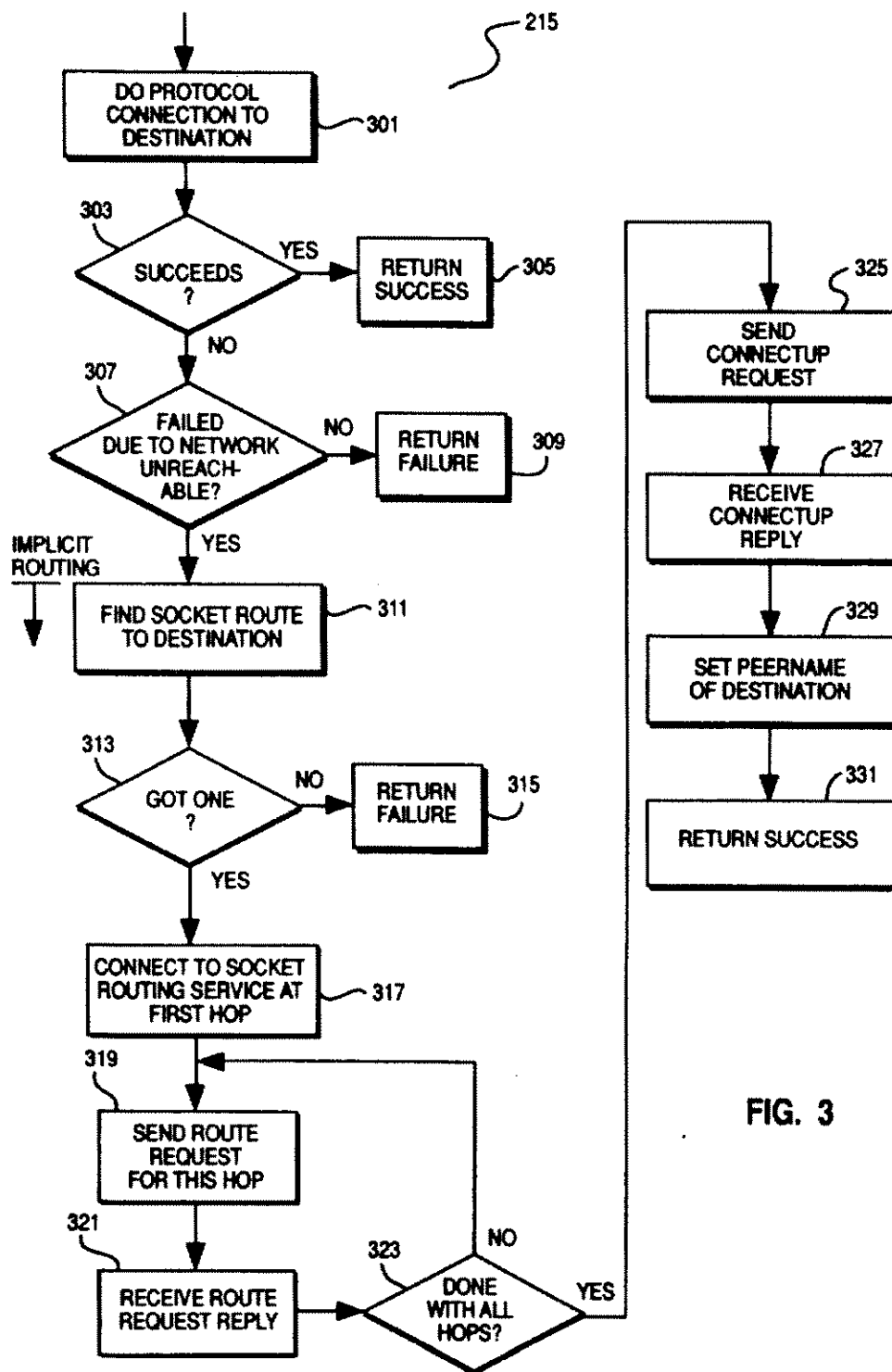


FIG. 1





U.S. Patent

Aug. 25, 1992

Sheet 4 of 7

5,142,622

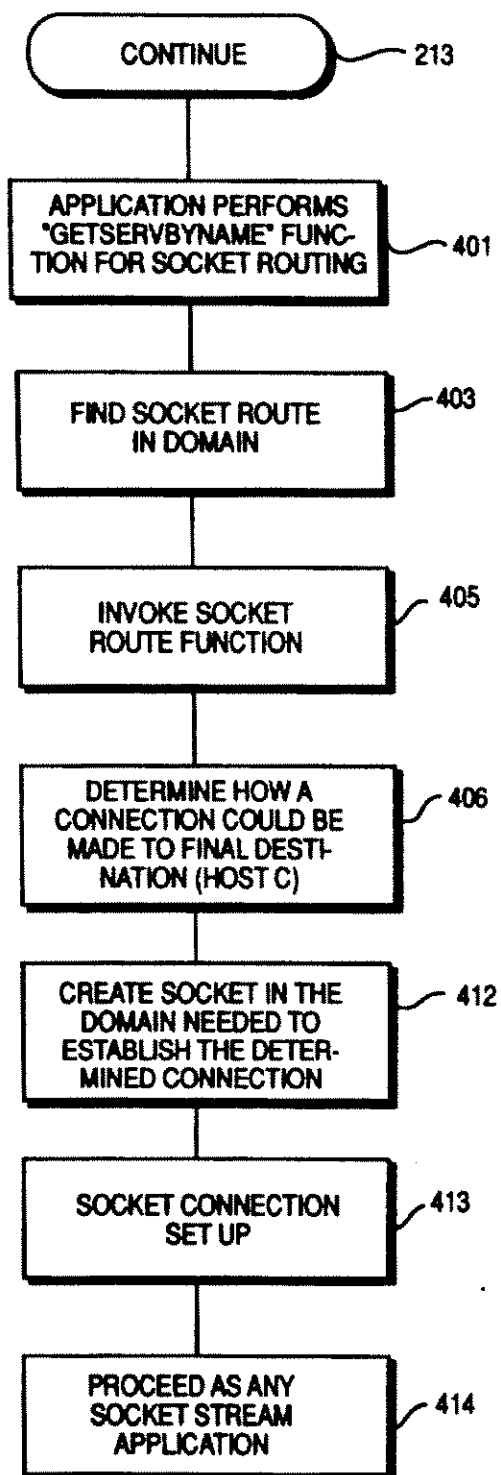


FIG. 4

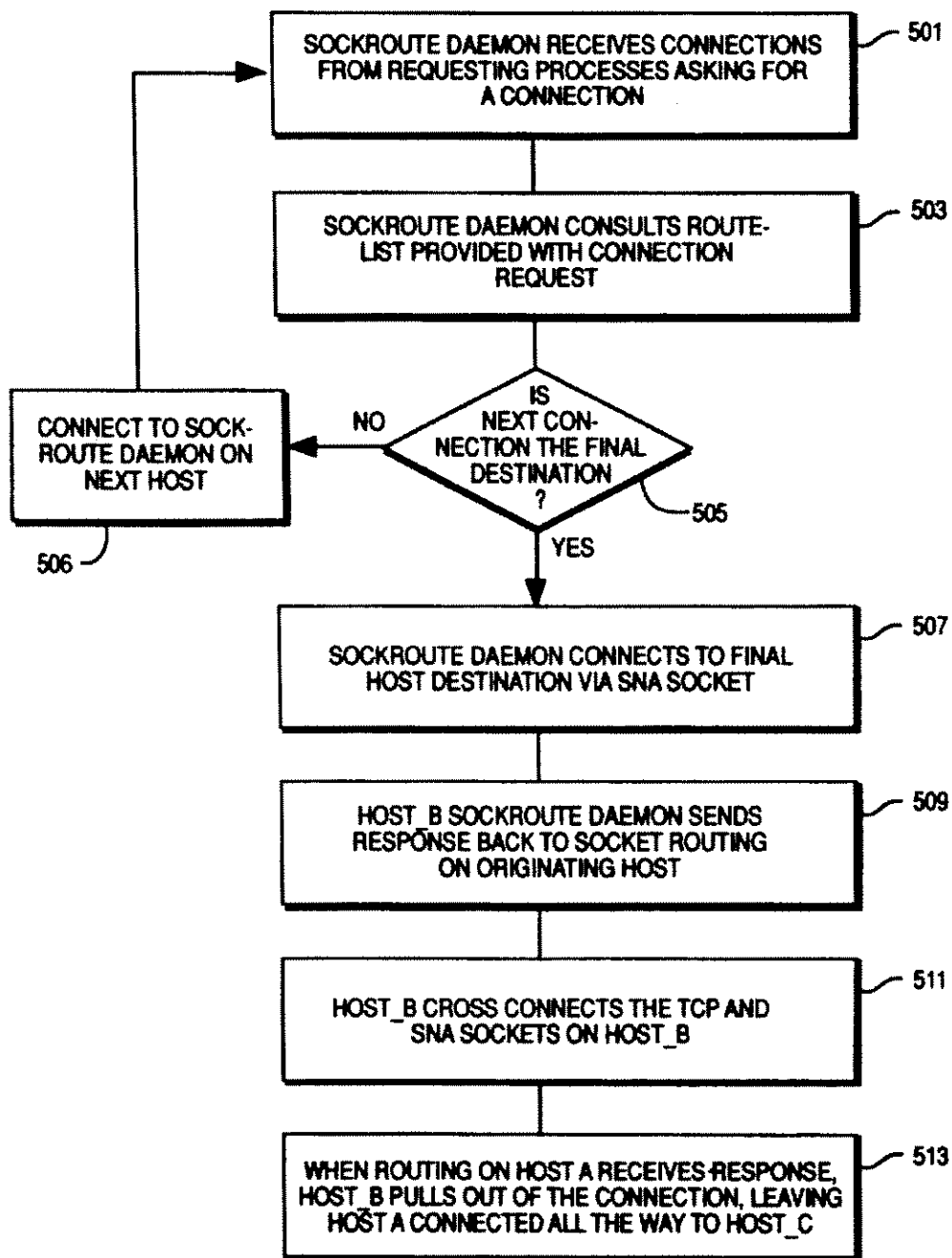


FIG. 5

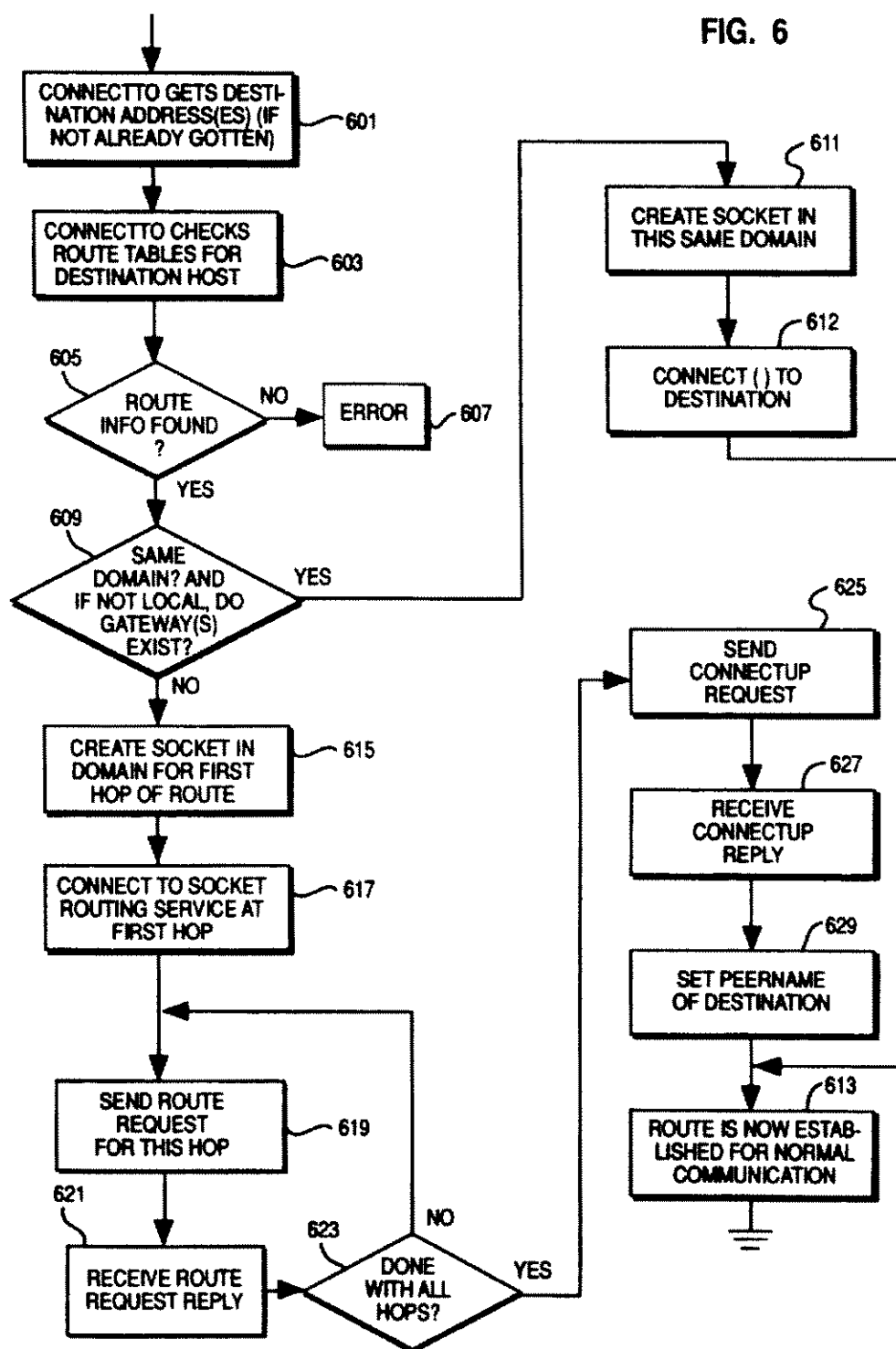
U.S. Patent

Aug. 25, 1992

Sheet 6 of 7

5,142,622

FIG. 6



U.S. Patent

Aug. 25, 1992

Sheet 7 of 7

5,142,622

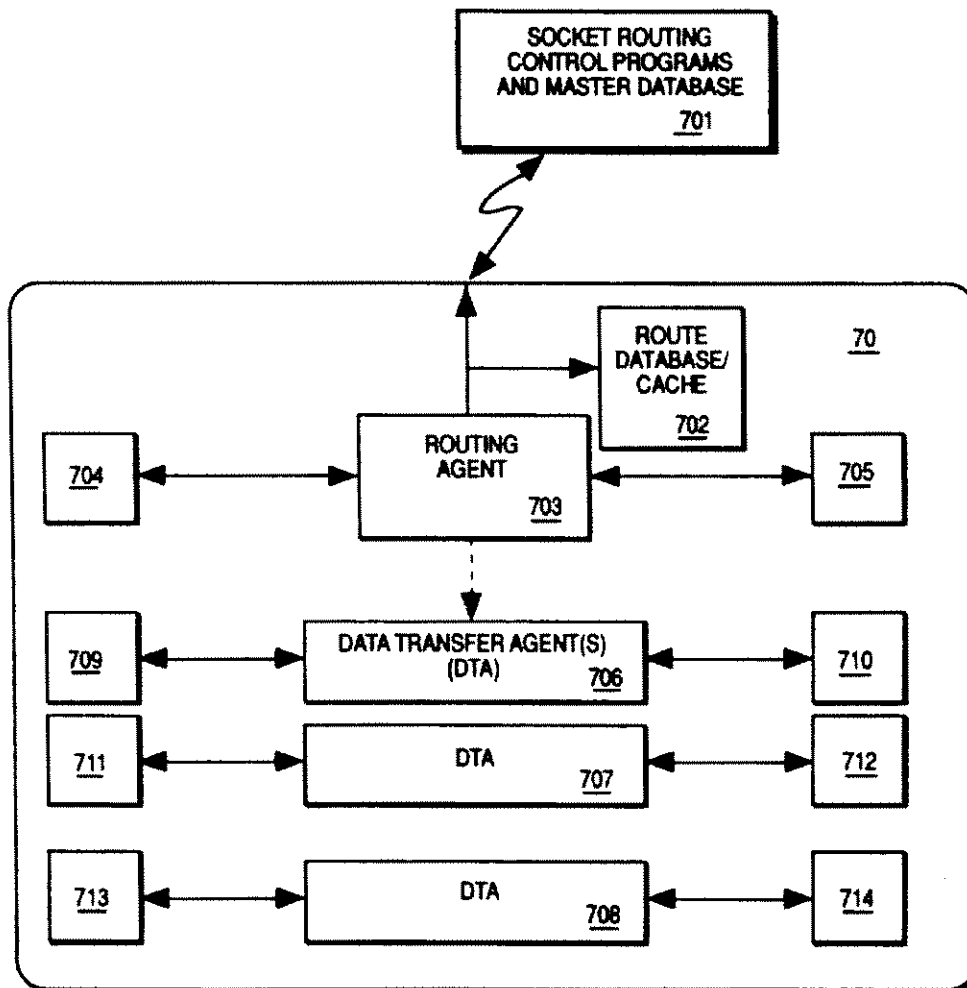


FIG. 7

5,142,622

1

SYSTEM FOR INTERCONNECTING APPLICATIONS ACROSS DIFFERENT NETWORKS OF DATA PROCESSING SYSTEMS BY MAPPING PROTOCOLS ACROSS DIFFERENT NETWORK DOMAINS

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to a network of data processing systems, and more specifically to the interconnection of a plurality of data processing systems between different network protocol domains, such as the different network protocol domains of SNA and TCP/IP.

2. Description of the Related Art

A system having multiple domains has at least one data processing system that is interconnected to at least two different data processing systems through at least two different network domains, i.e. network protocol architectures. A problem with multiple domains is the difficulty in allowing communication between machines which are connected to another type of network. For example, a data processing system utilizing SNA LU 6.2 as its network protocol can not automatically communicate with another data processing system utilizing TCP/IP as its network protocol. Both SNA LU 6.2 and TCP/IP are examples of stream protocols where data flows as a stream of indeterminate lengths, and the bytes are delivered in the correct order. The problem is routing a stream of bytes from a data processing system that utilizes a reasonably equivalent protocol, such as a stream protocol, to another data processing system that also utilizes a reasonable equivalent protocol, such as the stream protocol of this example, but wherein the two protocols are not

the exact same protocol, such as SNA LU 6.2 and TCP/IP.

It is known to solve the above problem at the application program level. An application program which is running on a data processing system at one end of the connection may be designed to utilize a specific network protocol. In this case, it is known to modify the application in order to reimplement the application to work over another protocol. This requires changing the source program code of the original application by some amount. Depending upon how the application program was originally designed, this may require a substantial amount of changes to the program code.

It is also known to solve the above problem by implementing the same protocol on both machines. For example, in order to use an SNA transaction application running in an SNA network, to apply transactions against data processing systems utilizing a TCP network, one could reimplement that transaction application against TCP by then putting TCP on the client data processing system, put IP over SNA, and gateway between the two. The client data processing system can then be implemented utilizing TCP/IP. The problem with this approach is having to reimplement the application to utilize the different protocol at one end of the

2

network or the other. This is especially burdensome if the application is large and complex.

There are some application level protocols that handshake back and forth over SNA, e.g. 3270 SNA. These have their own data format with meta-data in the data stream. There are other application level protocols, such as Telnet over TCP, that talk back and forth that have meta-data and data in the data stream. However, one can not get these two to talk together since these two have different data and meta-data in their data streams.

If an application utilized one protocol, and that application were to run on a data processing having a different protocol, knowing the data stream format, one could write the client half of the application on the data processing system utilizing the other protocol.

Therefore, in order to extend network connectivity, it is known to reimplement the application to utilize the different protocol, put one protocol on top of the other, and gateway between the two. It is also known to build a larger network utilizing each type of protocol through replication and duplication.

The term "sockets" is an application program interface (API) that was developed for the Berkeley version of AT&T's UNIX¹ operating system for interconnecting applications running on data processing systems in a network. The term socket is used to define an object that identifies a communication end point in a network. A socket can be connected to other sockets. Data can go into a socket via the underlying protocol of the socket, and be directed to appear at another socket. A socket hides the protocol of the network architecture beneath a lower layer. This lower layer may be a stream connection model (virtual circuit), or a datagram model (packet), or another model.

¹UNIX is licensed and developed by AT&T. UNIX is a registered trademark of AT&T in the U.S.A. and other countries.

A stream connection model refers to a data transmission in which the bytes of data are not separated by any record or marker. A virtual circuit implies that there appears to be one communications end point connected to one other communications endpoint. When the connection is established, only those two end points can communicate with each other.

Sockets are typed by domain (address family or network type), and model type (stream, datagram, etc.). If needed, the socket can be further specified by protocol type or subtype. The domain specifies the addressing concept utilized. For example, there is an internet IP domain, and also a SNA domain for networks utilizing TCP and SNA, respectively. As used herein, the word "domain" is used to refer to the address family of a socket, and not to a domain-naming domain. A domain-naming domain is a concept of a related group of hierarchical addresses, wherein each part of the address is separated by a delimiter, such as a period.

Since a socket is specified by the domain, sockets do not allow cross domain connections. This means that if an application program creates a socket in the Internet (Darpa) domain, it can only connect to sockets in that same domain. Note: "Darpa" is used to specify that Internet, short for internetworking, is not only used herein both to generically specify the internet layer of a particular protocol family which contains means for forwarding, routing control, and congestion control, etc., but also as a name for a particular implementation of an internet called the Internet or the Darpa Internet, or the Arpa Internet. Another name for this internet

5,142,622

3

layer is the Internet Protocol (IP). TCP/IP is also commonly used to refer to this protocol.

Originally, the requirement that a socket can only connect to sockets in the same domain was a reasonable restriction. This simplified the program code when there was only one really useful domain anyway. With the advent of the usage of other domains (specifically SNA), cross domain connections have become desirable. For example, cross domain connections would allow mailers to transport mail among domains. Also, cross domain connections would allow programs to communicate using the existing communication networks.

SUMMARY OF THE INVENTION

It is therefore an object of this invention to automatically route connections between data processing systems that utilize different protocols, independently of said applications running on said data processing systems.

It is a further object of this invention to route, at the socket level, between two networks when a cross-domain connection attempt is detected.

It is a further object of this invention to facilitate the interconnection between data processing systems by allowing socket based applications to easily span across different networks.

It is a further object of this invention to communicate between data processing systems in which one of the data processing systems utilizes TCP/IP and the other data processing system utilizes SNA.

It is a further object of this invention to communicate between two data processing systems via a third data processing system utilized as a TCP to SNA gateway.

It is a further object of this invention to communicate through a connection between two data processing systems both utilizing TCP on each of their local Internets, by bridging the network connection with a long haul SNA connection.

The system and method of this invention automatically routes a connection between data processing systems, independently of an application running on the data processing systems, having different network domains. The preferred embodiment describes the cross domain interconnections with reference to the different network domains of TCP (transmission control protocol) and SNA (systems network architecture).

The routing is automatically performed at a layer which contains the communication end point objects. In the AIX² operating system, and other operating systems based upon the Berkeley version of the UNIX operating system, this layer is called the socket layer.

²Trademark of IBM Corporation

An intermediate processing system is utilized to gateway between a processing system utilizing a network domain such as TCP, and another processing system utilizing a different network domain such as SNA. Alternatively, the client data processing system can be implemented utilizing TCP/IP which can then be gatewayed through socket routing on the same machine into an SNA data stream without an intermediate processing system performing the socket routing.

In any event, the socket layer which performs the socket routing contains facilities to automatically route a connection across different domains.

In the client processing system which is attempting to create a connection, a socket is created in a particular domain. If the socket is in a different domain, the socket does not fail if the socket routing facility of this inven-

4

tion is implemented. The connect function is modified to catch the attempts at a cross domain connection. If a connect function is attempted on a socket in a different domain, then the socket routing facility of this invention is invoked.

Alternatively, a connectto function can be implemented which takes the place of and combines the functions of the socket function and the connect function. With the connectto function, a socket is not created until the route is known. This alleviates the unnecessary work of creating a socket which may fail, and then performing actions as a result of the failed socket. The connectto function determines how a connection can be made, and then creates a socket in the domain that is needed to establish the determined connection.

Through either of the above approaches, a connection to a socket in a different domain can be made through an intermediate socket. When data arrives from one end of the connection to the intermediate socket, the intermediate socket immediately sends the data to the other end of the connection instead of queuing the data for process intervention at the intermediate processing system.

In addition, if the intermediate socket is queried for the address of the other end of the connection, the intermediate socket identifies the connecting host as opposed to the intermediate host. In this way, the socket routing facility of the intermediate host is transparent to the hosts at each end of the connection.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 is a diagram showing a connection from a process AA on host A to a process CC on host C. Socket routing is utilized to cross the boundary between the networks of type A and type C at host B.

FIG. 2 is a flow diagram showing the operational scenario of FIG. 1 using explicit and implicit routing.

FIG. 3 is a flow diagram showing the modified steps in performing a connect () function to a destination.

FIG. 4 is a flow diagram showing the steps of creating a socket if the host does not have a socket in the specified domain.

FIG. 5 is a flow diagram showing the steps performed at host B.

FIG. 6 is a flow diagram showing the steps of a connectto () function.

FIG. 7 is a more detailed diagram of the socket routing facility of this invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The following description describes an architecture for routing virtual circuits based on sockets. Although this implies stream sockets, the invention is not limited to stream protocols or to sockets. The concepts of this invention could be applied to similar communication end points that are utilized within other operating systems.

Referring to FIG. 1, a process AA, 10, in a data processing system 11, host A, desires to connect its socket facilities 12 to the process CC, 40, in a data processing system 41, host C. The data processing system 11 is shown as only supporting a particular domain of sockets AF_A, 13, such as TCP, and data processing system 41 is shown as only supporting sockets that exist in the domain having address family C, 43. Since the naming conventions and the underlying transport mechanisms

5

5,142,622

are different between address family A, 13, and address family C, 43, no interconnection can take place without an intermediate facility. The intermediate facility is the socket routing facility 70 in socket layer 32, which exists in data processing system 31, shown as host B.

To describe the initiation of a connection, the process AA, 10, in the data processing system 11, will activate a connection through the sockets programming interface to the general socket code, 12, which in turn goes through the address family specific socket code for AF_A, 13. The necessary data and control information will be handled by the interface and physical access layers, 14. The data will then go out on the network 50 and end up going into data processing system 31, shown as host B, via the interface layer 34, and then through the code for address family 30 A, shown as AF_A, 36.

For comparison, data processing system 21, shown as host D, shows existing internet routing within a single address family, the address family A, AF_A, 23. It should be noted that the cross connection occurs within the address family A, 23. Almost any TCP/IP implementation can route within its own address family. Likewise, SNA has similar gateway and forwarding capabilities. The cross over as shown in data processing system 21 is independent of the model type of either stream or datagram. It is only dependent upon being within the same network domain.

In data processing system 31, the connection request packets will go through the interface layer code 34 to the address family A code, AF_A, 36, through the general socket layer 32, and into the socket routing code 70. The socket routing code facility 70, is where the address mapping and cross connection takes place. The cross connection arrows 37 are shown drawn in the socket routing layer 70 of data processing system 31, as opposed to the cross connection arrows 27 which are shown in the address family code 23 of data processing system 21.

A connection request generated in the socket routing code 70 of data processing system 31 will then go down through the address family C code, AF_C, 33, and through the interface layer code 35 for the other network 60, such as SNA. The connection request packets go across the network 60 to the interface layer code 44, up to the address family C code, AF_C, 43, continuing through the general socket interface layer code 42 where the connection is registered. Then the process CC, 40, can respond to the connection request in order to establish the connection between cross domain networks.

FIG. 7 shows item 70 of FIG. 1 in greater detail. Item 701 is the programs and data for controlling the socket routing facility. A connection request to establish socket routing will come in on the sockets for this service, items 704, and 705. The routing agent software, item 703, will accept the connection, which creates a data socket, items 709-714. The route request message will come in on that data socket, and the routing agent, 703, will consult its route database, 702, to see if a route is possible. If a route is possible, the routing agent, 703, will consult its route database, 702, on how to establish the route. Then, the routing agent creates a matching data socket (item 710 for item 709, etc.), and connects to the next hop. When the routing agent software receives any replies for further route hops, it forwards them back to the socket routing requestor via the accepted data socket. When all hops are made, the socket routing agent will create a data transfer agent, items 706-708,

6

that joins the pairs of data sockets, and forwards data from one to the other and vice versa.

The above scenario is further described in the following programming design language code. The following includes examples and uses programs and function names to describe the operational scenario of FIG. 1. The following operational scenario assumes a telnet (or similar program) connected to a remote processing system that is separated by at least one domain boundary. The following uses three machines: "host_A" is connected to "host_B" via TCP, and "host_B" is connected to "host_C" via SNA.

```

/*from application view/*
user on host_A says "telnet host_C"
telnet does a gethostname for "host_C"
telnet tries to create a socket for domain of "host_C"
- it fails
telnet does a getservbyname for sockroute
- it finds (the only) sockroute available in TCP
domain
telnet invokes sockroute function to get which domain
to initiate the connection in (or to get a route
to host_C)
since telnet knows it is now using socket routing it
uses the (initial domain and routelist) to
1. create a socket in its initial domain. (TCP)
2. connects to sockaddr of "host_C" telnetd
-or "connectto routelist telnetd"
when socket connect succeeds, proceed as any
SOCK_STREAM app would
- alternatively ( with connectto() as "full function")
user on host_A says "telnet host_C"
telnet does a gethostname (or getaddrbyname) for
"host_C" - to see if it exists and to get
host_C's address
telnet does a "connectto ( host_C:telnetd,
SOCK_STREAM) - which gets a connected socket.
COPYRIGHT IBM CORPORATION 1988

```

The above program design language code is further explained with reference to FIGS. 2-4. The term "telnet" is a remote terminal emulator having the argument "host_C". This invokes the terminal emulator to a remote host, which in this case is "host C", step 201, FIG. 2. "Gethostbyname" is a function call of the telnet program which gets the addressing information for host C, step 203, FIG. 2. The addressing information for host_C will include a domain and an address within the domain.

At this point, the routing can be performed either explicitly or implicitly. Explicit action would involve the user code invoking a router function, if the initial attempt to create a socket fails. Implicit action would simply be doing a connectto () on the destination address. In explicit routing, the advantage is explicit control by the application. The disadvantages are lack of centralized control, and more complicated user code. In implicit routing, the advantages and disadvantages are just the opposite of those stated above. In implicit routing, the advantages are more centralized control, and less complicated user code. In implicit routing, the disadvantage is that the application does not have direct control.

With explicit routing, Telnet tries to create a socket within that domain, step 204. If the host does not have sockets of that domain, step 205, the socket creation will fail, step 211. At this point, the application, Telnet, invokes a router function, step 213 FIG. 4, if the socket attempt failed, step 211. If the host does have sockets within this domain, the socket attempt will succeed, step 206. If the socket attempt succeeds, the application

5,142,622

7

does a connect (), step 215. The connect () is further shown with reference to FIG. 3. If the connect () succeeds, step 217, FIG. 2, the communication between the two processes proceeds as is typically known in the art, step 207.

If a connect in the same socket domain failed, then (possibly with a socket option set) the socket routing would be invoked. This provides implicit routing, FIG. 3, even in the case of a connection between two domains of the same type, using an intermediate domain of different type.

As shown in FIG. 3, modifying the function connect () enables the connect () to catch those situations in which socket routing is needed to gateway between two like domains using unlike domains. If a normal connection, step 301, fails, step 303, and the failure is due to the destination network being unreachable, step 307, then an attempt at implicit routing will be made. This begins with step 311 where a socket route is sought for the destination. If no route is found, then an error is reported, step 315. If a route is found, a connection is made to the socket routing service at the first hop, step 317. Then, a route request is sent, step 319, and the route request replies are received, step 321, until all the hops are connected, step 323. At this time, a connect up request is sent to tell all of the routers to set up the line for data transmission, step 325. After the connect up reply is received, step 327, the peer address of the destination is set for the local socket, step 329, and an indication of success is returned to the invoker of this connect, step 331.

Referring back to FIG. 2, a connectto function can be added to the generic socket layer code to implement implicit routing from an application level, step 221, FIG. 2. The connectto function is called instead of a socket function and a connect function. The function of the socket system call and the function of the connect are combined into the connectto function. The advantage of this is that the connectto function can handle more addressing issues. Also the connectto function does not need to create a socket in the kernel, which may fail, and then have to act upon the failed socket.

The socket parameters of the connectto function would include the type and the protocol. Since the previous connect call has arguments for the host name, the connectto function would take the name of the host in a more portable form, such as the name of the host in a text stream, whereas, connect takes the name of the host in a socket structure.

Referring to FIG. 6, the connectto () function is further described. If connectto () is implemented so that it takes a host name as an argument, then it gets the destination address, step 601. Using this address, the function checks the route table for the destination, step 603. If no route is found, step 605, then an error is returned, step 607. If the destination is in the same domain, and no unlike domains are required for gateways, step 609, then a socket is created in the same domain, step 611. A normal connection is established to the destination, step 612. The route for communication is then established, step 613.

If the destination is not the same domain or unlike domains are required for gateways, step 609, then a socket is created in the domain of the first hop, step 615. A connection to the socket routing service at the first hop is then established, step 617. A route request is sent, step 619, and a reply to the request is received, step 621, until all hops are connected, step 623. After this, a con-

8

nect up request is sent, step 625, and its reply is received, step 627. The peername of the destination is set for the local socket, step 629. The route is now available for normal communications, step 613.

With the following modifications, referred to as socket routing, the creation of a socket can continue, step 213, as shown in FIG. 4, when the host does not have a socket in the specified domain, step 205, FIG. 2. The modifications take place at the client side, host_A. Host_C is referred to as the server.

The telnet application performs a "getservbyname" function for the socket routing service, step 401, FIG. 4. If, for example, the host only has sockets in the TCP domain, telnet will find the only socket route available in the TCP domain, step 403. Next, telnet uses the sockroute function, step 405, to determine the route and what domain of socket to create, step 406. Then, the socket is created for the initial hop of the route, step 412, and then the connection would be set up, step 413. At this point, the application can talk to the host as it otherwise would have with any other socket stream, and in this case, using the telnet data stream, step 414.

Assuming the route initialization is done by a daemon or library function on host_A (and not kernel code), then host_A's socket code doesn't really have much to do with socket routing. Basically, if socket routing is performed outside of the operating system kernel on host_A, then no changes to host_A's socket code need to be made.

The following programming design language code, and the following description with reference to FIG. 5 describes what happens on host_B.

```

/* on host_B */
sockroute daemon receive connection from host_A
(asking for connection to host_C)
sockroute daemon consults route table -or route list
provided with connection request.
sockroute daemon decides to connectto to host_C via
SNA socket
(since it is last hop, it doesn't need to connect
to a sockroute daemon on host_C)
when connection completes, host_B sockroute daemon
1. sends response back to socket routing on
host_A
2. cross connects the TCP and SNA sockets on
host_B
when routing on host_A receives response, it pulls out
of the way, leaving telnet connected all the way to
host_C

```

COPYRIGHT IBM CORPORATION 1988

Essentially, the above code describes the scenario in which a service waits around for a connection. With reference to FIG. 5, the sockroute daemon, which runs on host_B, receives connections from other processes requesting its services, step 501. The sockroute daemon is analogous to a telephone operator who is requested to make a connection to another person from a caller. The requesting process, caller, supplies the sockroute daemon, operator, with the necessary connection information in order to make the connection, step 503. Once the sockroute daemon makes the connection, the sockroute daemon leaves the connection. If this connection leads to the final destination, step 505, no other sockroute daemons on a next host need to be called, and the sockroute daemon connects to the final host destination via a SNA socket, step 507. However, it is possible to have multiple sockroute daemons, operators, that are needed to make a connection from a first host to a final host

9

destination. If this connection does not lead to the final host connection, then another sockroute daemon on a next host must be called, step 506, and the above steps repeated.

The sockroute daemon on host_B then sends a response back to the socket routing service on the originating host, host_A, step 509. Host_B cross connects the TCP and SNA sockets on host_B, step 511. When the routing service on host_A receives the response, host_B pulls out of the way. This leaves a telnet connection all the way from host_A to host_B, step 513.

It should be noted that since host_C is the end of the line, its socket layer is entirely unaffected for data transfer purposes.

There is a function called `getpeername()` that is part of the sockets programming interface. A socket can also be queried as to which service is connected to it. For example, if host_C queried its socket to determine which service at the other end it was connected to, the response would be the intermediate host, host_B, instead of the actual service at the other end of the connection which in this example is host_A. Therefore, the `getpeername` would need input from the socket routing code at both ends of the connection, as well as some kernel changes, for it to work in a transparent fashion. For transparency, the `getpeername` would respond with host_A, the real end of the completed connection, if the socket in host_C was queried as to the party at the other end of the connection.

The details of the address mapping and socket routing facilities within the socket layer 32, which effectuates the cross domain connections, are described hereafter.

Gatewaying of socket based protocols is achieved by looping two sockets together at the top end. Such a mechanism would allow a router to create a path that would cross domain boundaries. A router in this context would be program code that would decide how to get to one data processing system to the other such as in the internet layer of TCP/IP. SNA also has similar code. The mechanism for looping two sockets together at the top end would not require file descriptors, or process switching time on the connecting node, once the connection is established. The following illustrates the changes to the socket layer interface of an operating system, such as the AIX operating system that utilizes the Berkeley sockets, that may be made to implement socket routing of this invention. These changes include the following:

- modify "connect" to catch cross domain connects
- add "connectto" to implement implicit routing from application level.
- as an option, create library functions for routing
- modify socket buffer handling, etc. to allow cross connections without process intervention
- as an option, add function so `getpeername` works transparently
- define socket routing protocol and messages (in kernel or as a daemon)
- if needed, modify nameserver for domain gateways and routing info.

If `connectto` is not used to hide the routing from the user in a library, it is also possible to create library functions to perform the routing. However, the user will require a facility to figure out which machine has a socket routing daemon to service an intermediary. These functions(s) would allow a user program to invoke socket routing with minimal effort. Possible function to be defined are:

5,142,622

10

"get_route"—user program asks for route (useable by `connect()`)

"get_type_of_socket_I_should_open_to_get_to_host"—done against the return from "get_route"—or does implicit "get_route".

`connectto`—(1) looks up route, (2) creates a socket in proper domain, (3) established connection.

i.e., instead of

`hp=gethostbyname(host);`

(fill in `sockaddr` from `hp...`)

`so=socket(AF_XX, SOCK_STREAM, 0); connect(so, sockaddr, sockaddrlen);`

a program does

`so=connectto(host, SOCK_STREAM, 0);`

In addition, modifying socket buffer handling will allow cross domain connections without process intervention. Previously, a socket is set up such that when data arrives, the data is stored in a queue while the data waits for a process to read it. At the gateway, the socket routing machine, when data arrives from one end of the connection, the data has to be automatically sent out the other side to the other end of the connection, and vice versa.

A current implementation of socket buffering would require that a process be running against all the sockets that are cross connected. A more efficient means would be to add this cross connection at a socket buffer layer, so that no process scheduling needs to be done to send the data on its way. In either case, flags are added to the socket data structures.

As previously mentioned, additional function is added to the "getpeername" function to enable the intermediate host to appear transparently in the connection between the originating host destination and the final host destination. Previously, the socket peer address has been handled by protocol dependent means. A change is required so that `getpeername()` works correctly. The change involves having the peer address propagated by the route daemons, in both directions. Then the routing code at each end of the connection would do a "set peer address" operation, which would override the protocol's peer address function.

The socket routing facility of this invention also requires a socket routing protocol and messages. It is desirable that the socket routing code handle routing in a flexible manner. To achieve this, a preferred embodiment of this invention has a socket routing daemon on each machine that is an interdomain gateway. The daemon would be listening on well-known socket(s) for routing requests. When a request came in (via a connecting socket) the routing daemon would examine the request and perform the desired action.

These requests (and their responses) are as follows:

Messages For Socket Routing Protocol

and the information that goes with each message

route request—sent to request a route be set up

originator address

hop destination address

flag for intermediate or final hop

route request reply—received to indicate completion and success/fail of route request

status for success or failure

connectup request—sent to establish normal data pathway

<none>

connectup reply—received to indicate completion and success/fail of connectup request

status for success or failure

11

The socket routing service code is used to perform routing at the intermediate nodes, i.e. the gateway node. When a request for service arrives at the gateway machine, such as for any other socket connection, the request for service would arrive at a particular socket which would be the socket of the socket routing daemon. The process with this particular socket open could be either in the kernel or running as a user level process.

Therefore, the socket routing service code can be created as a daemon or in the kernel. Preferably, the socket routing service code will exist mostly or completely as a daemon. Some minor parts, such as ioctls (input output controls) to tie sockets together, may exist as part of the kernel. However, these minor parts support the daemon, and are not really a part of the socket routing service code. As an alternative, it is also possible to put the routing implementation part (as opposed to the route figuring out part) in the kernel, which would save process context switch time.

Another modification may be made to implement the socket routing of this invention. The nameserver may be modified for domain gateways and routing information. The (name) domain name server needs to have a type of data for inter(socket) domain gateways. It may also be desirable for it to find gateways when looking up a host address. It would be desirable if it would flag the fact that a host requires an inter(socket) domain gateway to get to it.

While the invention has been particularly shown and described with reference to a preferred embodiment including sockets, the underlying idea of cross domain connections could be achieved with other operating systems having other communication endpoints other than sockets. It will be understood by those skilled in the art that various changes in form and detail may be made without departing from the spirit and scope of the invention.

I claim:

1. A system for communicating between a first data processing system in a first network domain and a second data processing system in a second network domain, wherein said first network domain has a network protocol architecture different from said second network domain, said system comprising:

at least one communication end point object in a layer of said first data processing system in said first network domain and at least one communication end point object in a layer of said second data processing system in said second network domain;

means, independently of an application running on either of said data processing systems, for automatically establishing, in said layer of said first data processing system and in said layer of said second data processing system, a connection between said first processing system and said second processing system and comprising means for mapping protocols between said first and second network domain; and

means for communicating over said connection between said first data processing system and said second data processing.

2. The system of claim 1 wherein the first network domain is a Transmission Control Protocol and the second network domain is a Systems Network Architecture.

3. A system for communicating between a first data processing system in a first network domain and a second data processing system in a second network do-

5,142,622

12

main, wherein said first network domain has a network protocol architecture different from said second network domain, said system comprising:

at least one communication end point object in a layer of said first data processing system;

an intermediate data processing system having at least one communication end point object in a layer of said intermediate data processing system;

at least one communication end point object in a layer of said second data processing system;

means, in said intermediate data processing system, for establishing automatically routed connections in said layer of said first data processing system, said layer of said second data processing system and said intermediate data processing system and comprising means for mapping protocols between said first and second network domain, said first and second processing systems each including means for executing respective application programs; and means for communicating through said automatically routed connections between said first data processing system in said first network domain and said second data processing system in said second network domain.

4. The system of claim 3 wherein said means for communication immediately sends any data received from one end of said routed connection to said other end of said routed connection.

5. The system of claim 3 wherein said first data processing system includes a socket layer of socket code in said first data processing system;

said at least one communication end point object in a layer of said first data processing system is a socket in said socket layer of said first data processing system;

said intermediate data processing system includes a socket layer of socket code in said intermediate data processing system;

said at least one communication end point object in a layer of said intermediate data processing system is a socket in said socket layer of said intermediate data processing system;

said second data processing system includes a socket layer of socket code in said second data processing system;

said at least one communication end point object in a layer of said second data processing system is a socket in said socket layer of said second data processing system.

6. A system for communicating between a first data processing system in a first network domain and a second data processing system in a second network domain, wherein said first network domain has a network protocol architecture different from said second network domain, said system comprising:

at least one socket in a socket layer of said first data processing system in said first network domain;

at least one socket in a socket layer of said second data processing system in said second network domain;

means, independently of an application running on either of said data processing systems, for establishing in said socket layer of said first data processing system and in said socket layer of said second data processing system an automatically routed socket connection between said first data processing system and said second data processing system and

5,142,622

13

comprising means for mapping addresses between said first and second network domain; and means for communicating through said socket connection between said first data processing system and said second data processing system.

7. A method for communicating between a first data processing system in a first network domain having a socket and a second data processing system in a second network domain, wherein said first network domain has a network protocol architecture different from said second network domain, said method comprising:

establishing, by said first data processing system, a socket in said second data processing system in said second network domain; and

invoking a routing facility to automatically establish a socket connection between said socket in said first data processing system and said socket in said second data processing system when said socket in said second data processing system is established and comprising means for mapping protocols between said first and second network domain;

communicating over said socket connection between said socket in said first data processing system in said first domain and said socket in said second data processing in said second domain; and executing an application program on each of said first and second processing systems.

14

8. An operating system for use with a plurality of data processing systems for communicating between a first data processing system in a first network domain and a second data processing system in a second network domain, wherein said first network domain has a network protocol architecture different from said second network domain, said operating system comprising:

at least one socket in a socket layer of said first data processing system in said first network domain;

at least one socket in a socket layer of said second data processing system in said second network domain;

means, independently of an application running on either of said data processing systems, for automatically routing, in said socket layer of said first data processing system and in said socket layer of said second data processing system, a socket connection between said first data processing system and said second data processing system and comprising means for mapping addresses between said first and second network domain;

means for establishing said socket connection; and means for communicating through said socket connection between said first data processing system and said second data processing system, wherein said data first and second processing systems each include means for executing respective application programs.

* * * * *

EXHIBIT B



US006377988B1

(12) **United States Patent**
Spector et al.

(10) **Patent No.:** US 6,377,988 B1
(45) **Date of Patent:** Apr. 23, 2002

(54) **CONTROLLING ELEMENTS OF A
TELECOMMUNICATIONS NETWORK**

- (75) Inventors: **Adam Spector**, London; **Paul Abraham**, Croydon, both of (GB)
- (73) Assignee: **British Telecommunications public limited company**, London (GB)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/836,301**

(22) PCT Filed: **Nov. 7, 1995**

(86) PCT No.: **PCT/GB95/02617**

§ 371 Date: **Jun. 11, 1997**

§ 102(e) Date: **Jun. 11, 1997**

(87) PCT Pub. No.: **WO96/15635**

PCT Pub. Date: **May 23, 1996**

(30) **Foreign Application Priority Data**

Nov. 10, 1994 (GB) 9422722

(51) Int. Cl.⁷ **G06F 15/16; G06F 15/177**

(52) U.S. Cl. **709/224; 709/209; 709/221**

(58) Field of Search **340/825.06, 825.07; 395/200.54, 200.55, 200.56, 200.68, 200.74; 379/220, 221; 709/224, 225, 226, 238, 244, 223, 220, 221, 209**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,419,667 A * 12/1983 Gurr et al. 340/825.06
4,611,320 A * 9/1986 Southard
5,008,879 A * 4/1991 Fischer et al. 370/401

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

WO WO A 93 18598 9/1993

OTHER PUBLICATIONS

Rabie, "Evolution to Multivendor Intelligent Network Management", International Switching Symposium 1992, Session C1, Paper 1, vol. 1, Oct. 25, 1992, Yokohama, JP, pp. 60-64, XP 000337617.

Liao et al, "Toward the Intelligent Integrated Network Management", Globecom '90, Session 802, Paper 6, vol. 3, Dec. 2, 1999 San Diego, US, pp. 1498-1502, XP000218826.

Cameron et al, "Integrated Network Operations Architecture and Its Application to Network Maintenance", IEEE Communications Magazine, vol. 25, No. 8, Aug. 1987, New York, US, pp. 48-53.

Garrison et al, "The BT Network Traffic Management System: a Window on the Network", British Telecommunications Engineering, vol. 10, No. 3, Oct. 1991, London, GB, pp. 222-229, XP000279042.

Hong et al, "Design and Implementation of A Generic DISTRIBUTED Application Management System", Globecom 93, vol. 1, Nov. 29, 1993, Houston, US, pp. 207-211, XP 000428055.

Kiriha et al, "An Automatic Generation of Management Information Base (MIB) For OSI Based Network Management System", Globecom 91, Session 19, Paper 5, vol. 1, Dec. 2, 1991, Phoenix, US, pp. 649-653, XP 000326045.

Primary Examiner—Mark H. Rinehart

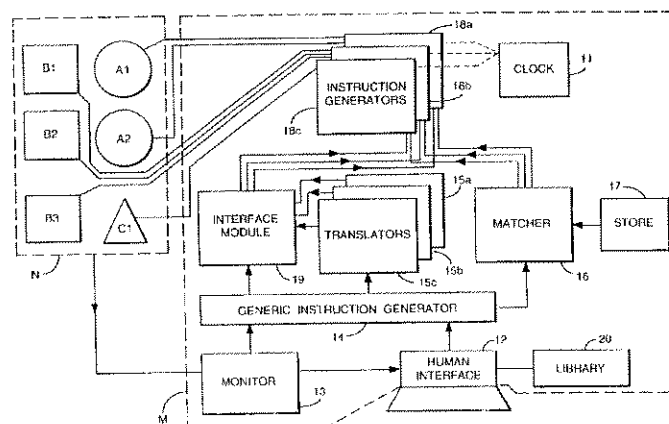
Assistant Examiner—Marc D. Thompson

(74) *Attorney, Agent, or Firm*—Nixon & Vanderhye P.C.

(57) **ABSTRACT**

A plurality of elements of a system, such as a telecommunications network are controlled by generating a generic instruction, in a generator selecting the element for which the instruction is directed, in a matcher and interface module and transmitting the instruction to the respective elements. By transmitting a generic instruction applicable to all relevant elements the operator of the system can control all the elements with a single instruction. A generic instruction can be translated in translators into separate instructions applicable to each individual element or defined groups of elements.

23 Claims, 5 Drawing Sheets



US 6,377,988 B1

Page 2

U.S. PATENT DOCUMENTS				
			5,420,916 A *	5/1995 Sekiguchi 379/230
			5,426,421 A *	6/1995 Gray 709/223
5,182,750 A *	1/1993	Bales et al.	379/221	
5,193,152 A *	3/1993	Smith 709/220		
5,237,561 A *	8/1993	Pyhälämmi		
5,317,742 A *	5/1994	Bapat 395/680		
				* cited by examiner

U.S. Patent

Apr. 23, 2002

Sheet 1 of 5

US 6,377,988 B1

Fig.1a.

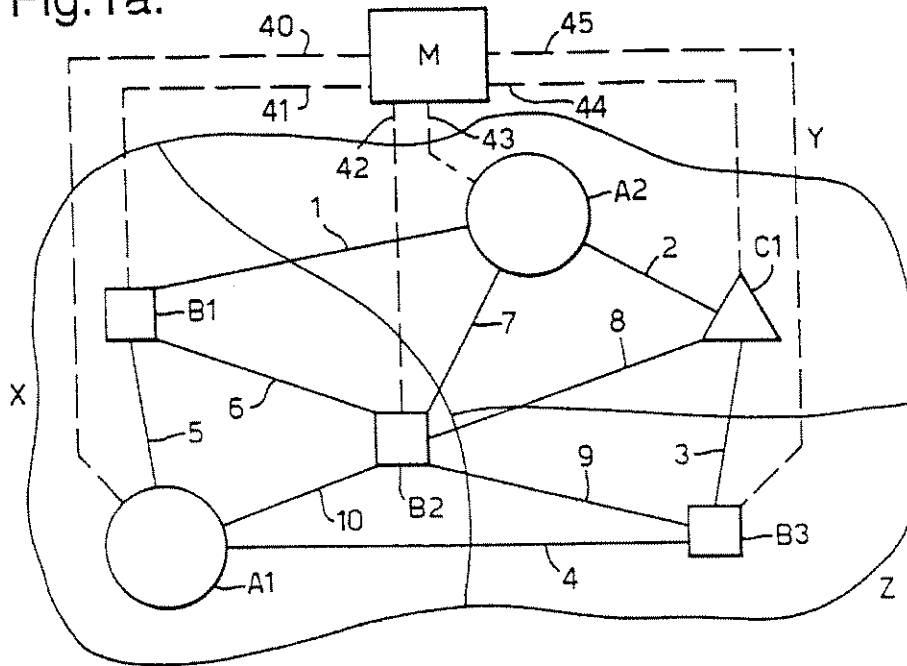
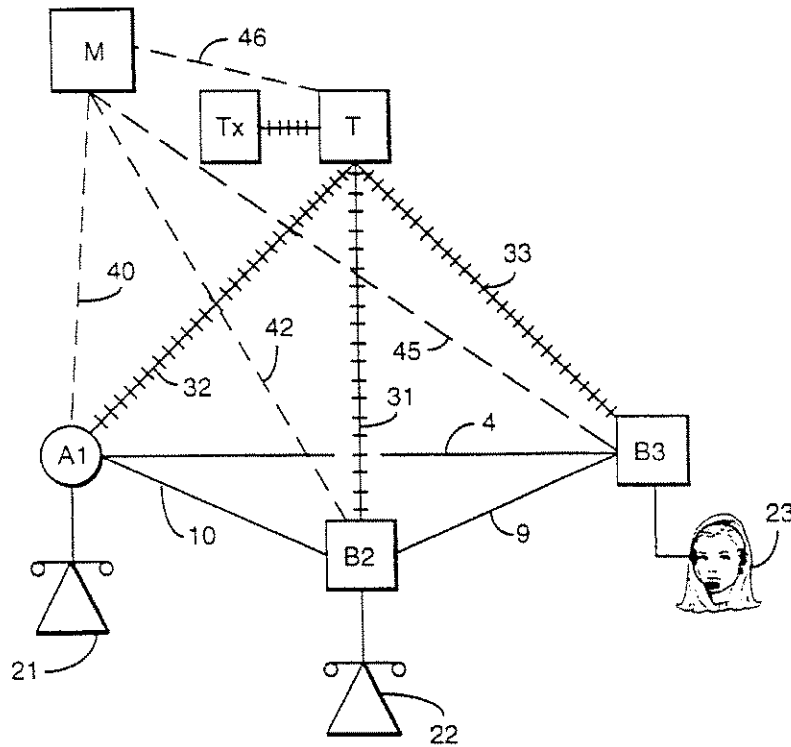


Fig.1b.



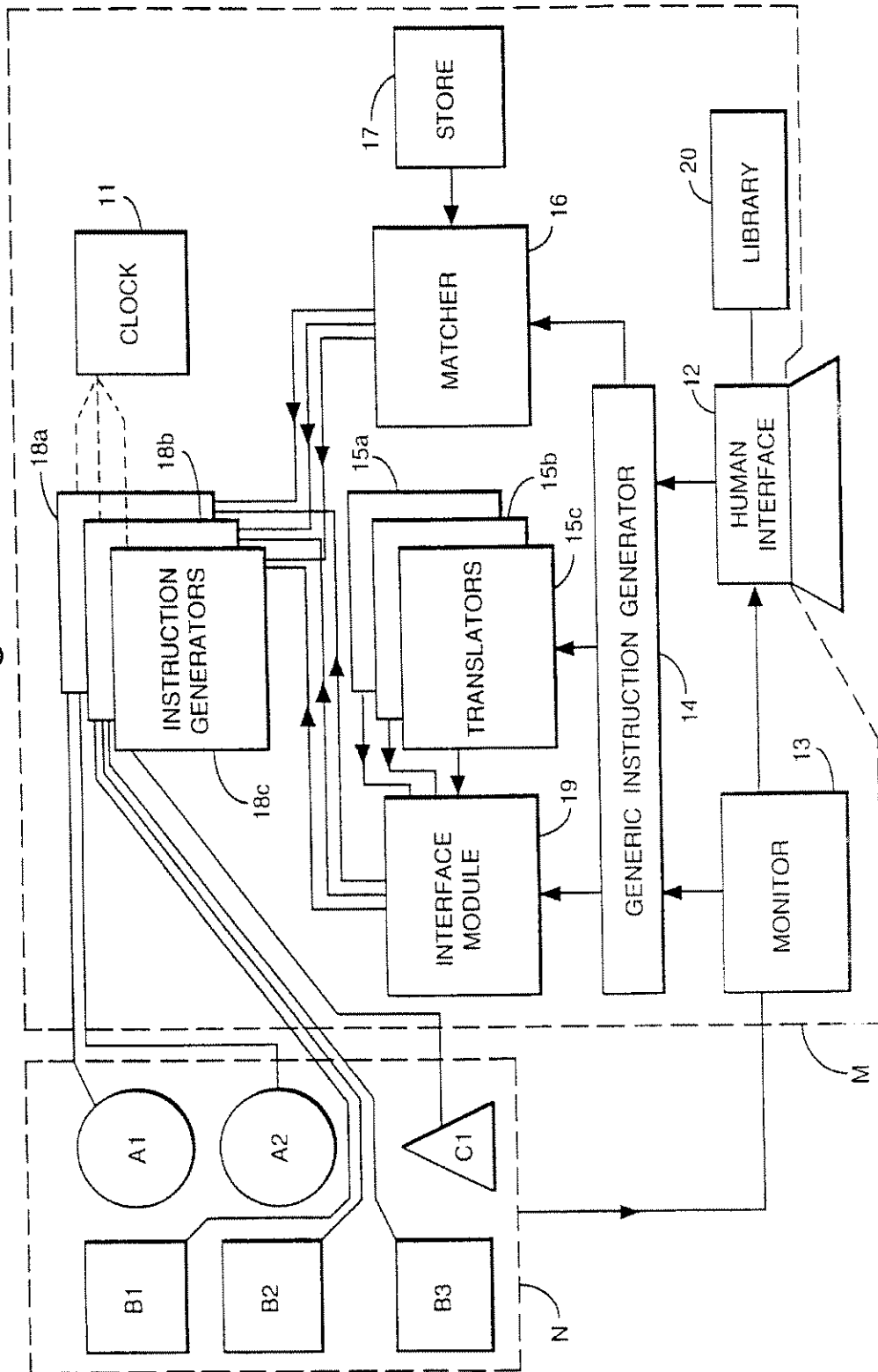
U.S. Patent

Apr. 23, 2002

Sheet 2 of 5

US 6,377,988 B1

Fig.2.



U.S. Patent**Apr. 23, 2002****Sheet 3 of 5****US 6,377,988 B1****Fig.3.**

SWITCH	LOCATION	TYPE
A1	X	A
A2	Y	A
B1	X	B
B2	X	B
B3	Z	B
C1	Y	C

Fig.4.

LINE	TERMINATIONS	
1	A2	B1
2	A2	C1
3	B3	C1
4	A1	B3
5	A1	B1
6	B1	B2
7	A2	B2
8	A2	C1
9	B2	B3
10	A1	B2

U.S. Patent

Apr. 23, 2002

Sheet 4 of 5

US 6,377,988 B1

Fig.5.

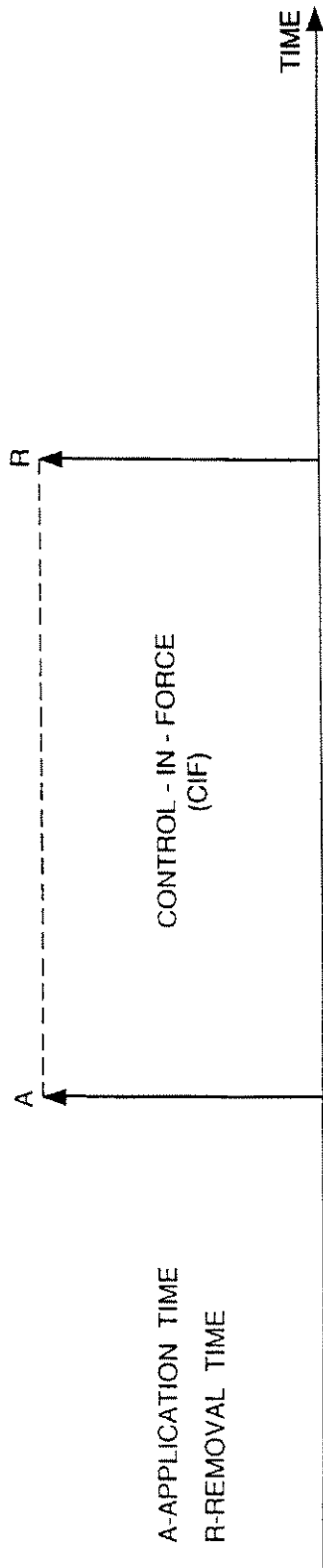
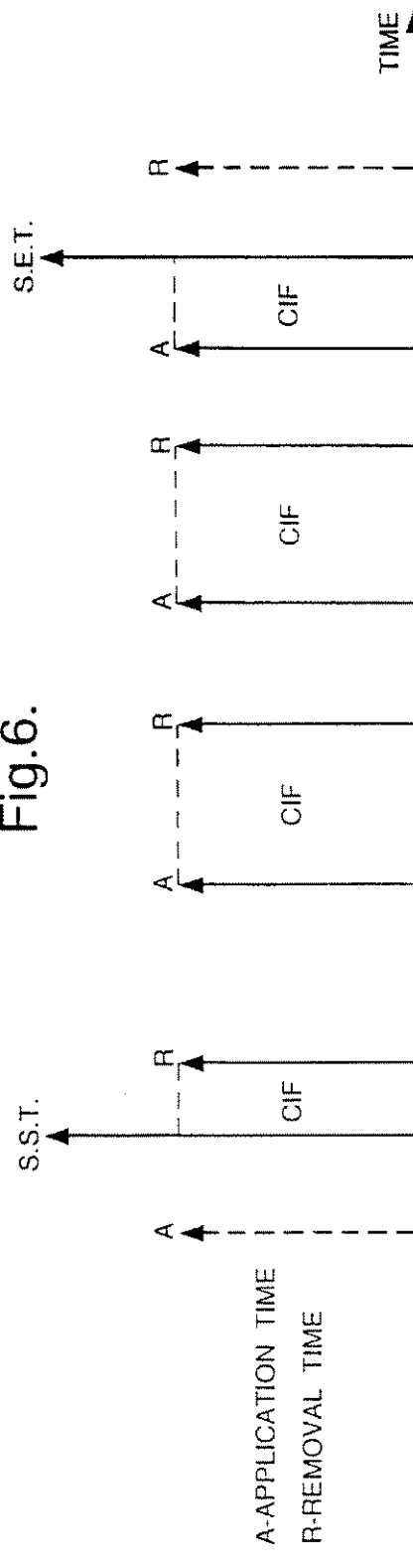


Fig.6.



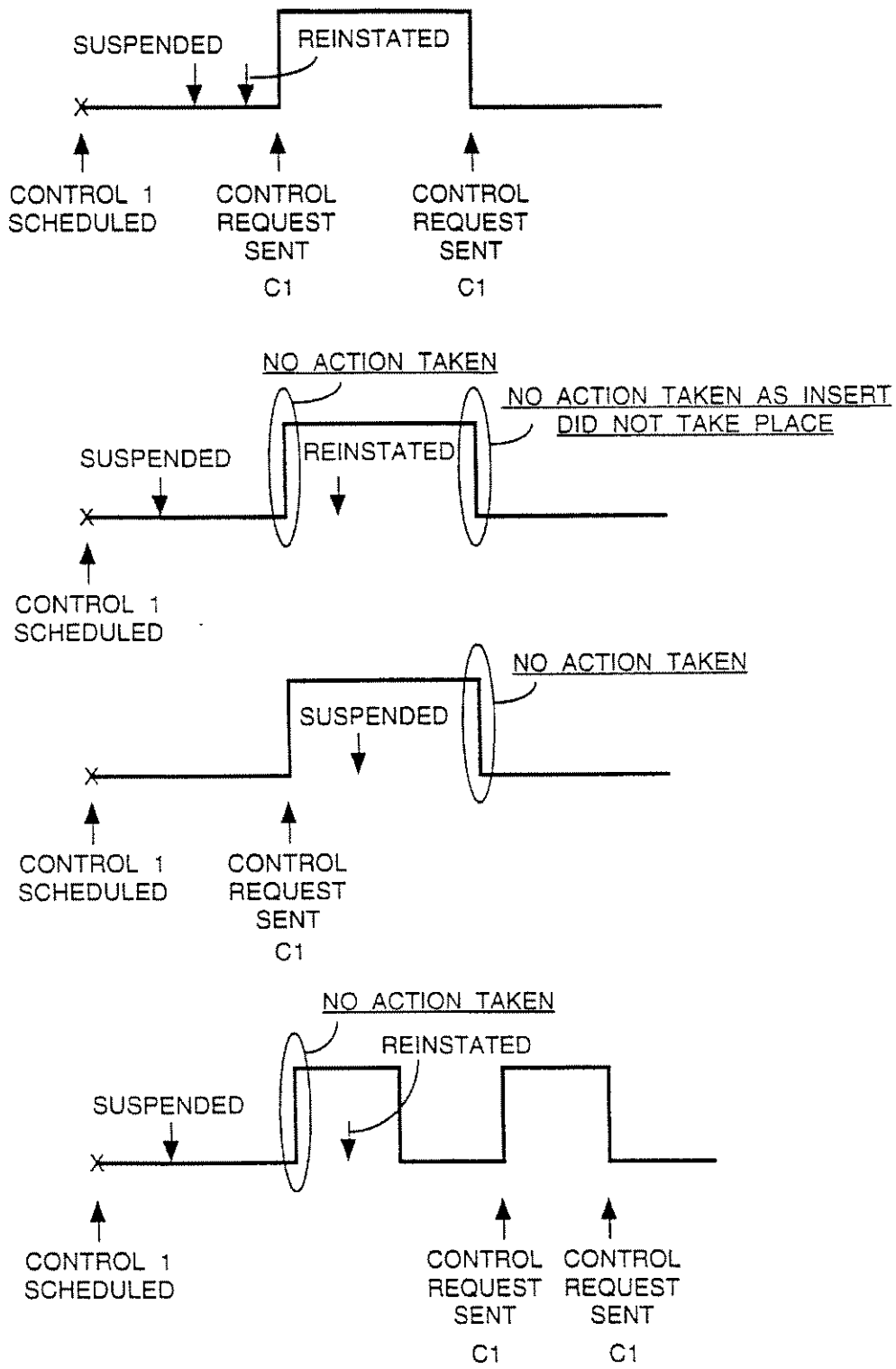
U.S. Patent

Apr. 23, 2002

Sheet 5 of 5

US 6,377,988 B1

Fig.7.



US 6,377,988 B1

1

**CONTROLLING ELEMENTS OF A
TELECOMMUNICATIONS NETWORK****BACKGROUND OF THE INVENTION****1. Field of the Invention**

This invention concerns controlling elements of a telecommunications network, particularly but not exclusively for traffic management.

2. Related Art

Although modern telecommunications networks are highly automated, they still require some monitoring and centralised control in order to deal with unusual circumstances such as overload conditions. A typical instance is a sudden surge in the number of calls made to a given telephone number, perhaps as a result of the number being given out on a television broadcast in an advertisement, during a phone-in show, or if an emergency contact number is given out on a news bulletin reporting a major accident or natural disaster.

The effect of such surges is to overload not only the line directly involved, but also the local exchange (also known as a switch) serving it. This results in calls to and from all subscribers served by that switch failing because all trunk lines serving the switch are busy with call attempts to the one number, most of which will fail.

The concepts of destination volume controls such as call blocking and call-gapping have been developed to overcome this problem. Call blocking arranges that a proportion of calls to a target number are failed by the originating exchange. Call gapping arranges that, after a call attempt is made to a target number, no further call attempts can be made to that number until a predetermined interval has elapsed. Both these systems reduce unnecessary use of the trunk network by failed call attempts.

A problem which arises in applying centralised control to a telecommunications network is that, in a typical network, exchanges are not identical. This is because in a developing network, at any given time, more than one type of exchange will be in use. Moreover, the different characteristics of the areas served by different exchanges may make different types of exchange appropriate in different locations. Consequently, certain functionality may only be available to certain exchanges or, even where the functionality is universal, individual instructions may be required to operate them. It is therefore necessary to tailor the instructions for each exchange. Furthermore certain services, such as call-gapping and call blocking, may be required only for a selected subset of exchanges, such as those serving the area in which the number has been broadcast.

It is known, for example from patent specification no. WO93/18598 (Nokia), to generate a command for a network element in a generic protocol which is translated into the appropriate protocol for the network element concerned. However it is necessary for the user of this system to transmit a command in the generic protocol for each element to be controlled. This can be time consuming and problematic, for example if several exchanges are required to co-operate and some of them do not have the necessary functionality.

SUMMARY OF THE INVENTION

According to a first aspect of the invention, there is provided a method of controlling a plurality of elements of a telecommunications network, comprising the steps of generating a generic instruction, selecting the elements for

2

which the instruction is directed, and transmitting the instruction to the respective elements.

According to a second aspect of the invention, there is provided a controller for a telecommunications network having a plurality of functional elements, comprising means for generating a generic instruction, and means for transmitting the instruction to the respective elements, characterised in having means for selecting the elements to which the instruction is directed.

By transmitting a generic instruction applicable to all relevant elements, the operator of the system can control all the elements with a single instruction. A generic instruction can be translated into separate instructions applicable to each individual element or defined groups of elements.

Certain elements may not have the ability to carry out certain functions, accordingly the method may provide that if one or more of the selected elements is not capable of performing the required instruction no instruction is transmitted to that element or elements. However, the ability of the network as a whole to carry out the desired functions may be dependent on the ability of each individual element to carry out a predetermined function. The method may therefore provide that, if any element is incapable of performing the required instruction, no instruction is transmitted to any of the elements. In other words, no instruction is transmitted to any element unless all the elements required to co-operate can carry out their individual instructions. Alternatively, certain network functions may be performable by parts of the network independently of the ability of other elements to do so, so the method may provide that if any elements are incapable of performing the instruction, the instruction is sent only to the elements which are capable of performing the instruction.

Different compatibility criteria may be used for different network functions.

In a preferred arrangement instructions may be prepared and stored for onward transmission in response to a predetermined condition. This allows the network to respond automatically and promptly to a condition such as a localised overload occurring in the system.

The predetermined condition may be the expiry of a pre-set time interval, allowing advance scheduling of network controls. For regular events, such as weekly or daily 'phone-in' programmes, the time interval may be re-set after each transmission of the instruction. However, for one-off situations such as special events the time interval is not re-set.

The instructions to the network elements may be arranged to cause the inhibition of a signalling function. For example, if calls to a single number are overloading the system, and that number is subject to a number translation process (for example converting a toll-free number to an exchange number), calls to that number can be gapped or blocked by inhibiting call set-up signals being sent to the number translation network element. This prevents abortive call set-up attempts clogging the signalling network, as well as freeing the traffic network itself.

In a preferred embodiment, the method comprises the steps of generating an instruction in high-level language, a selection pattern output, and an interface type message, converting the high-level instruction into instructions in formats compatible with each of a respective element type; and comparing the selection pattern data with stored information to select the elements to which the instruction is to be sent.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described by way of example, with reference to the figures, in which:

US 6,377,988 B1

3

FIG. 1a shows a simple telecommunications network controlled by 3 network manager, illustrating one arrangement according to the invention;

FIG. 1b is a diagram of a telecommunications network having a separate signalling network, illustrating a further arrangement according to the invention;

FIG. 2 shows the functional components of a network manager embodying the present invention and their relationship with a number of switching centres;

FIGS. 3 and 4 show simplified details of the store element 17 of FIG. 2;

FIG. 5 shows an illustration of a single shot schedule operated according to the invention;

FIG. 6 shows an illustration of a multi-shot schedule operated according to the invention; and

FIG. 7 shows how such scheduled controls can be implemented according to the invention.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENT

FIG. 1a shows a simple telecommunication network having six switches (exchanges) A1, A2, B1, B2, B3, C1 all operating under the centralised control of a network manager M. The switches are interconnected by traffic-carrying links (1, 2, 3, 4, 5, 6, 7, 8, 9, 10; solid lines) and each switch is connected to the network manager M by a respective control link (40, 41, 42, 43, 44, 45; dotted lines).

FIG. 1b shows a variant of FIG. 1a. Only three exchanges A1, B2, B3 are shown for simplicity, connected, as in FIG. 1a, by a traffic network 4, 9, 10 and also by a signalling network 31, 32, 33 to a signalling element T which includes a number translator Tx. The exchanges A1, B2, B3 and signalling element T are under the control of the network management centre M, by means of links 40, 42, 45, 46 respectively. Each exchange A1, B2, B3 is connected to a number of user terminals (only one shown in each case, 21, 22, 23 respectively).

The functional components of the network manager M which embody the invention, and their relationship to the network are illustrated in FIG. 2. It includes a store 17 which stores various attributes relating to the switches and the links between them. Such attributes include the types of equipment installed at each switch (in this example identified as type A, type B and type C which may for example be the System X, AXE 10 and 5ESS types currently in use by British Telecommunications plc) and the geographical location (in this example identified as area X (centres A1, B1, B2), area Y (centres A2, C1) and area Z (centre B3)). These areas may be defined by any suitable criterion or combination of such criteria. For example, a television coverage area may be appropriate for defining the switches to be controlled for call-gapping purposes when a phone-in number is broadcast, whilst for an enquiry line for a utility service such as transport, power, water etc. call-gapping may be required according to that utility's regional structure. More than one such geographical overlay may be defined. The data stored in store 17 may also include data on the current state of the elements, updated by input from the network elements.

The construction of network managers is well known to those skilled in the art. Typically a network manager is embodied by a computer whose software provides the necessary functional components.

The network manager M has three types of input, namely a time indication from a clock 11, a human interface 12 and a monitor 13 which receives inputs from the network N and

4

either transmits a message to the operator by means of the human interface 12 or generates an automatic response from the network manager. The human interface 12 is associated with a network traffic control library 20 in which may be stored standard input formats.

A generic instruction generator 14 receives inputs from the human interface 12 and the monitor 13. The generic instruction generator 14 provides three outputs: an instruction in high-level language which is passed to a group of translators 15a, 15b, 15c, a selection pattern output which is passed to a data matcher 16, and an interface type message transmitted to an interface module 19. The translators 15a, 15b, 15c are each configured to convert the high-level instructions received from the generic instruction generator 14 into an instruction in a format compatible with a respective switch type A, B or C. The data matcher 16 receives selection pattern data from the generic instruction generator 14 and compares this data with the information stored in store 17 in order to select the switches to which the instruction is to be sent. The data stored in the store 17 can be updated in response to inputs from the network N through the monitor 13.

FIGS. 3 and 4 show the organisation of data in the store 17.

FIG. 3 shows the data stored for the switches A1, A2, B1, B2, B3, and C, by geographical area and type of switching equipment. As well as these inherent characteristics, the store 17 may also hold data relating to the current condition of the switches.

FIG. 4 shows similar inherent data held for each link (1 to 10) in the system. This data includes the termination points and the capacity of each link.

An instruction generator 18a, 18b, 18c is provided for each type of switch A, B, C. Each instruction generator 18a, 18b, 18c receives data from the matcher 16, the clock 11 and, via the interface module 19, the respective translator 15a, 15b or 15c, from which instructions are generated for transmission to appropriate elements of the network N.

The instructions may be transmitted by the instruction generators 18a, 18b, 18c immediately, or at one or more predetermined times under the control of the clock 11. For example, the instructions may be transmitted to the network N on a regular basis, such as at a predetermined time and day of the week to coincide with the broadcast times of a phone-in television or radio show.

The interface module 19 ensures that a generic instruction which requires the co-operation of two or more switches is compatible with all the switches involved. The interface module 19 controls the operation of the instruction generators 18, 18b, 18c in the event that the instruction required is incompatible with the capabilities of one or more types of switch A, B or C. Such an incompatibility would be identified by the respective translator 15a, 15b or 15c, which transmits an error message to the interface module 19 should the instruction not be executable. The generic instruction generator 14 transmits an interface-type message to the interface module 19 which controls the operation of the instruction generators 18a, 18b, 18c. The interface-type message can be one of three types:

- i) AND: all switches to which the instruction is addressed must be capable of executing the instruction: if any switch is not so capable the instruction is not carried out at all
- ii) OR: any switch to which the instruction is addressed and which is capable of executing the instruction does so, even if other switches cannot do so.
- iii) CONDITIONAL: an instruction to control routes between switches is executable only in respect of those

US 6,377,988 B1

5

routes for which all switches controlling that route are compatible with the instruction.

A message is transmitted to the human interface 12 to indicate whether, or to what extent, the instruction can be executed.

The operation of the network manager M will now be described with reference to FIGS. 1a and 2. In this illustrative example, the function to be controlled is call-gapping, to be initiated at a set time once a week or in response to an overload condition detected by the monitor 13. For the sake of this illustration, it will be assumed that call-gapping is only required for switches in geographical areas X and Y, and that equipment of type A is not capable of providing call-gapping functionality.

The human operator provides an input through the human interface 12 to the generic instruction generator 14. This input identifies the function required (call-gapping), the proportion of calls to be gapped, the geographical area to be covered (in this example zones X and Y), and the time that the function should operate (for example with immediate effect until manually cancelled, or from 1900 to 2030 every Tuesday, or in response to an external stimulus such as an overload detected by the monitor 13).

The generic instruction generated in the generator 14 is transmitted to the bank of translators 15a, 15b, 15c which each translate the instruction into a form which can be handled by a respective switching system type A, B, C. In this case type A is incapable of performing call-gapping so an error message is generated by the translator 15a.

The generic instruction generator 14 has two further outputs. Firstly, it transmits the selection criteria to the matcher 16. In this case, the switches to be call-gapped are selected by geographical area. The selection of the set of switches to be selected may be done on the basis of the intrinsic architecture of the network, e.g. a set may comprise a DMSU (digital main switching unit) and its daughter DLEs (digital local exchanges). Alternatively the set may be manually selected, in order to meet a non-intrinsic criterion such as the coverage area of a television station, or the administrative regions of a utility company. These criteria are compared in the matcher 16 with the stored details held in the store 17, and the identities of the switches which meet the criteria are transmitted from the matcher 16 to the bank of instruction generators 18a, 18b, 18c. In the present example, the identities of the switches A1 and A2 are transmitted to the instruction generator 18a, the switches B1 and B2 to the instruction generator 18b, and the switch C1 to the instruction generator 18c. The identity of the switch B3 is not transmitted to instruction generator 18c because it is in zone Z and therefore does not meet the geographical requirement.

The other output from the generic instruction generator 14 relates to the type of instruction, and is transmitted to the interface module 19. This output identifies to the interface module 19 whether the operation should be partially executed even if some switches lack the necessary functionality. In the case of call-gapping or call blocking the operation can be partially executed in this way, so the interface module 19 transmits the instructions generated by the translators 15b and 15c to the respective instruction generators 18b, 18c, but transmits no instruction to the instruction generator 18a as only an error message was received from the translator 15a.

If the generic instruction cannot be carried out unless all relevant switches can co-operate, the arrival of an error message from any one of the translators 15a, 15b, 15c prevents the interface module 19 from transmitting any instruction to any of the instruction generators 18a, b, c.

6

If the generic instruction requires a route to be established between a first switch having the required functionality and a second switch lacking that functionality the route is not established. However, this does not prevent the first switch establishing routes to other switches in accordance with the generic instruction, provided these other switches have the required functionality.

The instructions generated in the instruction generator 18b, are transmitted to the switches B1 and B2, and the instruction generated in the instruction generator 18c is transmitted to the switch C1. The instructions are transmitted under the control of the clock 11, at the times programmed by the generic instruction generator 14.

Thus, by the input of a single instruction by the human operator or automatically through the monitor 13 the required function can be performed by all switching elements having suitable functionality within the geographical area specified, without the human operator needing to generate separate instructions for each type of switch or to specifically identify the switches to be controlled.

In the arrangement shown in FIG. 1b the system is configured so that the signalling network 31, 32, 33 is prevented from overloading with unsuccessful call attempts. Call set-up signals are carried over the signalling network 31, 32, 33 in order to set-up the necessary connections in the traffic network 4, 9, 10 between the exchanges A1, B2, B3. The user 23 is subject to a number translation service carried out by a number translator Tx in the signalling element T. In other words, the users 21, 22 can call the user 23 using a special number (e.g. a toll-free number) which is translated by the translator Tx to control the signalling element T to set-up a call to the user 23.

If an overload of calls using the special number is observed or predicted, the network manager M can control the switches A1, B2, B3 to inhibit the transmission over the signalling network 31, 32, 33 of call requests to that number. This releases capacity not only in the traffic network 4, 9, 10, but also in the signalling network 31, 32, 33. If one of the switches (e.g. A1) does not have the capability to inhibit the transmission of such call requests, they can still be inhibited by the network manager M at the signalling element T. This does not prevent unsuccessful call requests appearing on the signalling link 32, between the exchange A1 and the signalling element T, but it does prevent them propagating over the rest of the network.

The normal sequence of operation of the system will now be described. It may be viewed as consisting of three distinct processes: control definition, control scheduling, and control implementation. Firstly, users define pre-planned controls (PPCs) by specifying the type of control (e.g. destination call gapping) and optionally the values of the control parameters and the control target.

Once defined, the PPC may be saved in the network traffic control library 20 for later use or may be scheduled.

The primary function of the library 20 is to provide a repertoire of PPCs which may be quickly retrieved to deal with network problems.

PPCs in the library 20 may be retrieved for control scheduling. Users schedule a PPC by selecting one from the library 20 and specifying the schedule details and if necessary the control target and control parameters may be added (if they are not already defined), or changed (if they are defined). The PPC in the library 20 is not affected. Once scheduled, the control is entered onto the list of currently scheduled controls.

Scheduled controls may be positively authorised at any time within N minutes of their due implementation (where

US 6,377,988 B1

7

N is a pre-defined parameter), otherwise the control will not be implemented. If authorisation is not given within the pre-defined time period specified by N a report is generated.

Authorised controls are automatically implemented (inserted and removed) according to the schedule attached to the PPC. Users can monitor the progress of controls implementation in real-time.

The network traffic control library 20 allows users to build up and maintain a repertoire of PPCs. This allows users to save time by having access to templates to commonly used controls (e.g. to deal with focused overloads) where only the control target is different. It also allows the accurate implementation of a prepared complex sequence of controls. Network problems which require the fast and accurate deployment of a number of controls can be dealt with promptly. The library 20 may also be used as a source from which to build new controls.

There are three types of library entries:

1. basic PPC (with a single exchange/route control target)
2. basic PPC (with an exchange set control target)
3. linked PPC

The control targets may be specified, or may be left for the user to select. A linked PPC consists of a sequence of PPCs. PPCs may be stored with all or only some of the control parameters and the control target blank. The network manager will check that a PPC is fully specified when it is selected for scheduling.

A basic PPC with a single exchange/route as a control target is the simplest type of control. A basic PPC with an exchange set as a control target is also referred to as a broadcast control. These two types of basic PPCs are only distinguishable when the control target is specified.

A broadcast control is a control which is applied to a number of exchanges (exchange set). A facility is provided for defining and maintaining a library of exchange sets.

A linked PPC is a composite control and is defined by linking together a number of basic PPCs in a sequence to form a linked PPC. When the linked PPC is implemented, each PPC in the chain is implemented sequentially starting with the first.

Exchanges can be grouped together into exchange sets. Commonly used exchange set definitions may be stored in store 17 and used when defining broadcast controls. These are referred to as static exchange sets.

Exchanges can also be grouped by exchange type into dynamic exchange sets. The following default dynamic exchange sets are provided:

- All DMSUs (Digital Main Switching Units: trunk exchanges)
- All DCCes (Digital Call Centre Exchange: tandem/ junction exchange—intermediate level)
- All DLEs (Digital Local Exchanges)
- All DDSCs (Digital Derived Services Switching Centre)
- All DJSUs (Digital Junction Switching Unit—intermediate level)
- All dependent level exchanges of DMSU (one set per DMSU)
- All dependent level exchanges of a DCCe (one set per DCCe)
- Dependent level exchanges include DLEs, DJSUs and DCCes.

A traffic control is scheduled to deal with a particular problem on the network. Users have two sources for PPCs when scheduling, namely PPCs stored in the network traffic control library; or by defining a control for the purpose, allowing users to schedule new controls rapidly without

8

following the normal sequence of storing a control definition in the library and then retrieving it. The network manager will perform the same sequence of operations but this will be transparent to the user.

A network traffic control is applied to an exchange in two stages. It is defined using the facilities of the network traffic control library, and the schedule to be applied at specified times. The control could be applied immediately or stored for later application.

Both single-shot and multi-shot (repeating) schedules are provided:

single-shot where only one insertion and removal is involved

multi-shot where multiple insertions (and removals) are involved.

Scheduled controls may be named in such a way as to link them to specific events, e.g. Christmas Day.

A schedule may be specified without a removal time.

The single-shot timing mechanism provides the opportunity to define a period during which the traffic control should be applied. This is achieved in terms of a control insertion time, and a control removal time.

FIG. 5 provides an illustration of the type of schedule that may be achieved using a single-shot schedule. The hatched area represents the time over which the control is in force. 'A' represents the application time, and 'R' the removal time.

A control may be scheduled without a specified removal time; such a control is effective indefinitely. Facilities are provided to add later a removal time to the schedule; immediate removal is also possible.

The control insertion time is specified as a specific time e.g. 10:30 Nov. 14, 1994

The control removal time is specified in the same format as the insertion time, e.g. 13:15 Nov. 14, 1994 or it may be specified as a delta time (i.e. the control duration): e.g. 002 02.45

The delta time in the example means remove the control after an elapsed time of 2 days 2 hours and 45 minutes.

A multi-shot (repeating) schedule is a single-shot schedule but with the addition of a schedule start and end time and some repeating criteria (e.g. on Monday of every week)

The facility of multi-shot schedules provides the opportunity to define a repeating traffic control application pattern. The application pattern is defined in terms of:

- a schedule start time;
- a schedule end time;
- a control inserting time (within the period specified by the schedule start and end time);
- a control removal time (within the period specified by the schedule start and end time); and
- repeat criteria.

FIG. 6 provides an illustration of the type of schedule that may be achieved using a multi-shot schedule. The hatched area represents the time over which the control is in-force.

It is possible for the control in-force period to span a midnight boundary (i.e. control removal occurs on a different day to control insertion). A control is removed when its associated schedule expires.

Note that the control is not applied until Schedule Start Time SST and is removed at Scheduled End Time SET, although these do not necessarily coincide with control insertion and removal times.

The scheduled start time SST, control insertion time A and control removal time R are specified in the same format as for the single shot control above.

The scheduled end time defines a time after which no further applications occur.

US 6,377,988 B1

9

The scheduled end time is specified in one of the formats specified for the control removal time above.

The repeating criteria are identified in terms of:

days of the week, or days of the month (but not both), and calendar months of the year in which the above applies.

In the simplest case, a multi-shot schedule specifies a schedule start and end time and the days of the week or days of the month in which the control insertions and control removals apply within the specified schedule start and end time.

A more specialised repeating pattern based on the above can be defined, by further identifying the months within the specified schedule start and end time in which the control insertions and control removals apply.

It is possible for schedules to expire automatically, or be forced to expire by operator action.

In the case of a single-shot schedule, the schedule automatically expires on removal. For multi-shot schedules, automatic expiry occurs on the schedule end time.

After a control from the network traffic control library 20 is scheduled, the network manager maintains details of the scheduled control and will implement the control as specified in the schedule.

A facility is provided to allow users to track and monitor the current list of scheduled controls on the network manager and to monitor which of the scheduled controls are in-force (active). FIG. 7 shows how scheduled controls are tracked by the network manager through the scheduled control and active control lists.

In the event of a scheduled control application failing, the application is periodically retried until it is successful or the end of the scheduling period has been reached. The number of retries and the retry interval are configurable parameters.

A message is sent to the human interface 12 if the network manager fails to implement a scheduled control after the maximum number of retry attempts or the end of the scheduling period has been reached.

The implementation of a scheduled control can fail for several reasons. Firstly there may be inconsistent scheduled requests or other reasons (failure within network manager). Secondly there may be communications failure.

There may also be communications failures, maintenance failures (e.g. failure to process a job request), or exchange failures (e.g. the exchange does not accept a control request from the network manager).

In the first two cases the network manager will continue to apply the retry logic as specified in order to attempt to successfully implement the control. However, in the event of an exchange failure the network manager will stop the retry logic as soon as an exchange fails to accept a control request.

Overlapping schedules can be allowed subject to any traffic control implementation rules defined in the network manager, as described below.

Before a control request is implemented the network manager checks if there is an existing control on the control target, and if so, the actual control requests issued by the network manager are modified. The rules specified below, are general rules to be applied when implementing controls.

The network manager may issue a real-time "control status read" request before applying a control.

For inserting a basic PPC (Single Exchange/Route and Exchange Set), the network manager checks for a control of the same type on the target (which may consist of members of an exchange set). If a control of the same type is already in force, and if the schedule creation date of the scheduled control is earlier than the in force control, the scheduled control is not implemented; otherwise the scheduled control is implemented.

10

The user is able to suspend scheduled controls, which may be authorised, prior to implementation. Suspended controls can be reinstated at any time, however a control will not be implemented if the scheduled insertion time has been exceeded. Multi-shot controls will be implemented until the next insertion time after the reinstatement.

Before inserting a Basic PPC (Single/Exchange/Route and Exchange Set) control the network manager may issue a request to read the real-time control status.

If a control is already in force on the exchange or the exchange resource already exists, but with different parameters to the control to be implemented, then a "change" exchange-specific command is issued; otherwise an "insert" exchange-specific command is issued.

The network manager may be arranged such that if it discovers a control of the same type and detects that it did not insert the discovered control, the network manager alerts users, updates the controls database and sends a message to the central system mailbox. The scheduled control will then only be applied if agreed by a user with the Control Authorisation access privilege, and otherwise will be removed.

When removing a Basic PPC (Single Exchange/Route and Exchange Set) the network manager checks for a scheduled control of the same type on the target (which may consist of members of an exchange set). If a control of the same type is scheduled and due to be enforced (but not implemented due to the application of the first rule above—see FIG. 7) the control is applied using the second rule above.

If no control of the same type is scheduled and due to be in-force, before implementing a remove control request the network manager issues a real-time control status read request.

If a control is already in-force on the exchange or the resource already exists a recover exchange specific command is issued.

For linked pre-planned controls, each PPC in the chain would only be implemented if the implementation of the previous PPC was successful. Implementation of constituent PPCs in the chain would be according to the rules specified above for PPCs. Removal of the PPCs in the chain is in the reverse order to that of insertion.

If one of the PPC constituents fails during implementation, then the effect of all previously implemented constituent PPCs in the chain would be reversed.

Scheduled controls which have been authorised are implemented by the network manager at the specified times. At the point of implementation, the network manager maps the generic control to an exchange-specific control if the generic control is a linked control then each individual control in the linked control is mapped to the exchange-specific format when the individual control is implemented. The network manager then transmits the control request(s) to the appropriate instruction generator 18a, 18b, or 18c for each exchange, and saves the corresponding control response(s) returned by the instruction generator 18. Finally, the network manager updates the internal control database with the new status of the control.

In general, a control may be inserted, removed, or cancelled. In addition control status read requests may be issued to obtain the status of a network resource prior to inserting or removing a control on it. The cancel request does not result in any control requests to the network communications manager.

Control responses are matched with the control requests to form a complete control transaction. Each control transaction corresponds to an insertion or removal of a scheduled control.

US 6,377,988 B1

11

Network traffic management controls are instructions sent to an exchange via a Network Communications Manager) to modify exchange resources, with the purpose of overcoming problems existing on the network (e.g. by causing traffic to be re-routed).

Each equipment type (e.g. System X Phase 3) manages a different set of resources to perform its switching operations. However the network manager attempts to modify exchange resources on different exchange equipment types with the intention of modifying traffic patterns in similar ways.

In order that the network manager can interwork with a number of exchange equipment types, and allow additional equipment types to be added as the system evolves, without the system growing increasingly complex, a set of exchange-type independent Generic Controls is defined.

Some exchange-specific controls only appear on one particular equipment type or do not fit into the generic set. Facilities to create, schedule and manipulate these non-generic controls can be provided so that specific features offered only by one system, or specifically designed for a special requirement of one installation, are not lost despite not being generic to the system as a whole.

Each generic control stored on the network manager M has a defined set of parameters. The controller sends exchange specific control requests to the network elements when a scheduled control is implemented. The translators 15 convert generic controls stored in the network manager to exchange-specific versions at the point of implementation via a set of mapping rules.

When a user creates a Destination Volume Control such as call blocking, the network manager stores the values for the generic parameters defined for the control and not values for the exchange specific parameters. The exchange specific parameters are calculated according to a set of mapping rules at implementation time. A separate set of rules is required for each target equipment type.

Exchange specific parameters which cannot be calculated or derived from the generic parameters are assigned default values by the network manager. If however, a user creates the generic instruction knowing that the target(s) of the control includes exchanges having additional functionality, then values for the exchange-specific parameters, such as traffic category and call disposition, may be specified at control creation time.

Mapping rules are defined for each exchange specific control on each target equipment type supported by the control M.

In mapping the generic set of parameters to the exchange specific set, the mapping process will use the nearest available value for an exchange specific parameter if the required value is not available.

Non-generic controls such as System X Loopback can be stored and manipulated in their exchange specific form in the network manager. At implementation time, no mapping process is performed by the network manager and the control details are sent directly to the switches for implementation on the exchange.

What is claimed is:

1. A method of controlling a telecommunications network, the network comprising at least two groups of elements, wherein at least some of the elements of at least two of the groups are capable of performing a predetermined function in response to a group-specific instruction, the method comprising the steps of:

generating a generic instruction, for performance by a selected group of the elements there being a plurality of the elements in at least one selected group;

12

converting the generic instruction to group-specific instructions appropriate to each group of elements in which there is at least one of the selected elements; and transmitting to each selected element the group-specific instruction appropriate to its group.

2. A method as in claim 1 further comprising the steps of: determining which selected elements are capable of performing the required instruction, and inhibiting the transmission of the instruction to all of the selected elements if any of said elements do not have the required capability.

3. A method as in claim 1 further comprising the steps of: determining which of the selected elements are capable of performing the required instruction, and transmitting the instruction only to that element or elements.

4. A method as in claim 1 wherein instructions are prepared and stored for onward transmission in response to a predetermined condition.

5. A method as in claim 4, wherein the predetermined condition is the expiry of a time interval.

6. A method as in claim 5, wherein the time interval is reset after each transmission.

7. A method as in claim 5, wherein the time interval is not reset after the transmission.

8. A method as in claim 4, further comprising the steps of: monitoring the telecommunications system for a predetermined operating condition, and generating the generic instruction if the said predetermined operating condition occurs.

9. A method as in claim 8, wherein the predetermined operating condition is a network overload.

10. A method as in claim 1 wherein the instruction causes the inhibition of a signalling function.

11. A method as in claim 1 further comprising the steps of: generating a generic instruction in a high-level language; generating a selection pattern, indicative of the elements to be controlled by the instruction;

generating an interface type message, indicative of whether any of the elements selected by the selection pattern are to carry out the instruction if other elements of those selected are incapable of performing the instruction;

converting the high-level instruction into one or more lower-level instructions, each having a format compatible with a respective group of the elements; and

comparing the selection pattern output and interface type data with stored information to select the elements in each group to which each low-level instruction is to be transmitted.

12. A method as in claim 11 further comprising the additional steps of:

generating a message indicating whether, if some of the selected elements are unable to execute the instruction, the elements which are able to execute the instruction should do so, and

inhibiting or executing the operation of the said elements in response to said message.

13. A controller for a telecommunications network, the network comprising at least two groups of elements, wherein at least some of the elements of at least two of the groups are capable of performing a predetermined function in response to a group-specific instruction, the controller comprising:

means for selecting a plurality of the elements, to which an instruction to perform the predetermined function is to be transmitted;

US 6,377,988 B1

13

means for generating a generic instruction, for performance by the selected plurality of the elements;

means for converting the generic instruction to group-specific instructions appropriate to each of plural groups of elements in which there at least one of the selected elements; and

means for transmitting to each selected element the group-specific instruction appropriate to its group.

14. A controller as in claim 13, in which the selection means comprises:

means for determining which groups, of those to which the selected elements are capable of performing the required instruction, and

means for inhibiting the transmission of the instruction to all of the selected elements if any of said groups of elements do not have the required capability.

15. A controller as in claim 13, in which the selection means comprises:

means for determining which of the selected elements are capable of performing the required instruction, and

means for transmitting the instruction only to that element or elements.

16. A controller as in claim 13, further including:

means for preparing and storing instructions, and

means for transmitting the stored instructions in response to a predetermined condition.

17. A controller as in claim 16, further comprising:

a timing means, the transmitting means being arranged to transmit the instructions under the control of the timing means.

18. A controller as in claim 17, further including:

means for resetting the timing means after each transmission.

14

19. A controller as in claim 16, further comprising:

monitoring means for monitoring the telecommunications system for a predetermined operating condition,

the transmitting means being responsive to the monitoring means to transmit the stored instructions if the said predetermined operating condition occurs.

20. A controller as in claim 19, the monitoring means being configured to detect network overloads.

21. A controller as in claim 13 further comprising:

a generic instruction generator for generating generic instructions in high-level language and for generating selection pattern data, indicative of the element to be controlled by the instruction;

a plurality of translators each configured to covert the high-level generic instructions generated by the generic instruction generator into an instruction in a format compatible with a respective element type; and

a data matcher for comparing the selection pattern data generated by the generic instruction generator with information stored in a store, in order to select the switches to which the instruction is to be sent.

22. A controller as in claim 21, the generic instruction generator also having:

means for generating a message indicating whether, if some of the selected elements are unable to execute the instruction, the elements which are able to execute the instruction should do so, and

means for inhibiting or executing the operation of the elements in response to said message.

23. A telecommunications network comprising a controller as in claim 13, in combination with one or more network elements, at least some of the network elements having:

means for isolating a network signalling function; and

means to activate the isolating means in response to an instruction transmitted from the controller.

* * * * *

JS 44 (Rev. 11/04)

CIVIL COVER SHEET

The JS 44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. (SEE INSTRUCTIONS ON THE REVERSE OF THE FORM.)

I. (a) PLAINTIFFS CISCO SYSTEMS, INC. and CISCO TECHNOLOGY, INC.	DEFENDANTS TELCORDIA TECHNOLOGIES, INC.
(b) County of Residence of First Listed Plaintiff _____ (EXCEPT IN U.S. PLAINTIFF CASES)	County of Residence of First Listed Defendant _____ (IN U.S. PLAINTIFF CASES ONLY) NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF THE LAND INVOLVED.
(c) Attorney's (Firm Name, Address, and Telephone Number) 302-658-9200 Jack B. Blumenfeld (#1014) Morris, Nichols, Arsht & Tunnell LLP 1201 N. Market St.; Wilm., DE 19899	Attorneys (If Known)

II. BASIS OF JURISDICTION (Place an "X" in One Box Only)	III. CITIZENSHIP OF PRINCIPAL PARTIES (Place an "X" in One Box for Plaintiff and One Box for Defendant)																
<input type="checkbox"/> 1 U.S. Government Plaintiff <input type="checkbox"/> 2 U.S. Government Defendant <input checked="" type="checkbox"/> 3 Federal Question (U.S. Government Not a Party) <input type="checkbox"/> 4 Diversity (Indicate Citizenship of Parties in Item III)	<table style="width: 100%;"> <tr> <th style="text-align: left;">PTF</th> <th style="text-align: left;">DEF</th> <th style="text-align: left;">PTF</th> <th style="text-align: left;">DEF</th> </tr> <tr> <td>Citizen of This State</td> <td><input type="checkbox"/> 1 <input type="checkbox"/> 1</td> <td>Incorporated or Principal Place of Business In This State</td> <td><input type="checkbox"/> 4 <input type="checkbox"/> 4</td> </tr> <tr> <td>Citizen of Another State</td> <td><input type="checkbox"/> 2 <input type="checkbox"/> 2</td> <td>Incorporated and Principal Place of Business In Another State</td> <td><input type="checkbox"/> 5 <input type="checkbox"/> 5</td> </tr> <tr> <td>Citizen or Subject of a Foreign Country</td> <td><input type="checkbox"/> 3 <input type="checkbox"/> 3</td> <td>Foreign Nation</td> <td><input type="checkbox"/> 6 <input type="checkbox"/> 6</td> </tr> </table>	PTF	DEF	PTF	DEF	Citizen of This State	<input type="checkbox"/> 1 <input type="checkbox"/> 1	Incorporated or Principal Place of Business In This State	<input type="checkbox"/> 4 <input type="checkbox"/> 4	Citizen of Another State	<input type="checkbox"/> 2 <input type="checkbox"/> 2	Incorporated and Principal Place of Business In Another State	<input type="checkbox"/> 5 <input type="checkbox"/> 5	Citizen or Subject of a Foreign Country	<input type="checkbox"/> 3 <input type="checkbox"/> 3	Foreign Nation	<input type="checkbox"/> 6 <input type="checkbox"/> 6
PTF	DEF	PTF	DEF														
Citizen of This State	<input type="checkbox"/> 1 <input type="checkbox"/> 1	Incorporated or Principal Place of Business In This State	<input type="checkbox"/> 4 <input type="checkbox"/> 4														
Citizen of Another State	<input type="checkbox"/> 2 <input type="checkbox"/> 2	Incorporated and Principal Place of Business In Another State	<input type="checkbox"/> 5 <input type="checkbox"/> 5														
Citizen or Subject of a Foreign Country	<input type="checkbox"/> 3 <input type="checkbox"/> 3	Foreign Nation	<input type="checkbox"/> 6 <input type="checkbox"/> 6														

IV. NATURE OF SUIT (Place an "X" in One Box Only)				
CONTRACT <input type="checkbox"/> 110 Insurance <input type="checkbox"/> 120 Marine <input type="checkbox"/> 130 Miller Act <input type="checkbox"/> 140 Negotiable Instrument <input type="checkbox"/> 150 Recovery of Overpayment & Enforcement of Judgment <input type="checkbox"/> 151 Medicare Act <input type="checkbox"/> 152 Recovery of Defaulted Student Loans (Excl. Veterans) <input type="checkbox"/> 153 Recovery of Overpayment of Veteran's Benefits <input type="checkbox"/> 160 Stockholders' Suits <input type="checkbox"/> 190 Other Contract <input type="checkbox"/> 195 Contract Product Liability <input type="checkbox"/> 196 Franchise	PERSONAL INJURY <input type="checkbox"/> 310 Airplane <input type="checkbox"/> 315 Airplane Product Liability <input type="checkbox"/> 320 Assault, Libel & Slander <input type="checkbox"/> 330 Federal Employers' Liability <input type="checkbox"/> 340 Marine <input type="checkbox"/> 345 Marine Product Liability <input type="checkbox"/> 350 Motor Vehicle <input type="checkbox"/> 355 Motor Vehicle Product Liability <input type="checkbox"/> 360 Other Personal Injury	PERSONAL INJURY <input type="checkbox"/> 362 Personal Injury - Med. Malpractice <input type="checkbox"/> 365 Personal Injury - Product Liability <input type="checkbox"/> 368 Asbestos Personal Injury Product Liability PERSONAL PROPERTY <input type="checkbox"/> 370 Other Fraud <input type="checkbox"/> 371 Truth in Lending <input type="checkbox"/> 380 Other Personal Property Damage <input type="checkbox"/> 385 Property Damage Product Liability	FORFEITURE/PENALTY <input type="checkbox"/> 610 Agriculture <input type="checkbox"/> 620 Other Food & Drug <input type="checkbox"/> 625 Drug Related Seizure of Property 21 USC 881 <input type="checkbox"/> 630 Liquor Laws <input type="checkbox"/> 640 R.R. & Truck <input type="checkbox"/> 650 Airline Regs. <input type="checkbox"/> 660 Occupational Safety/Health <input type="checkbox"/> 690 Other LABOR <input type="checkbox"/> 710 Fair Labor Standards Act <input type="checkbox"/> 720 Labor/Mgmt. Relations <input type="checkbox"/> 730 Labor/Mgmt. Reporting & Disclosure Act <input type="checkbox"/> 740 Railway Labor Act <input type="checkbox"/> 790 Other Labor Litigation <input type="checkbox"/> 791 Empl. Ret. Inc. Security Act	BANKRUPTCY <input type="checkbox"/> 422 Appeal 28 USC 158 <input type="checkbox"/> 423 Withdrawal 28 USC 157 PROPERTY RIGHTS <input type="checkbox"/> 820 Copyrights <input checked="" type="checkbox"/> 830 Patent <input type="checkbox"/> 840 Trademark SOCIAL SECURITY <input type="checkbox"/> 861 HIA (1395ff) <input type="checkbox"/> 862 Black Lung (923) <input type="checkbox"/> 863 DIWC/DIWW (405(g)) <input type="checkbox"/> 864 SSID Title XVI <input type="checkbox"/> 865 RSI (405(g)) FEDERAL TAX SUITS <input type="checkbox"/> 870 Taxes (U.S. Plaintiff or Defendant) <input type="checkbox"/> 871 IRS—Third Party 26 USC 7609
REAL PROPERTY <input type="checkbox"/> 210 Land Condemnation <input type="checkbox"/> 220 Foreclosure <input type="checkbox"/> 230 Rent Lease & Ejectment <input type="checkbox"/> 240 Torts to Land <input type="checkbox"/> 245 Tort Product Liability <input type="checkbox"/> 290 All Other Real Property	CIVIL RIGHTS <input type="checkbox"/> 441 Voting <input type="checkbox"/> 442 Employment <input type="checkbox"/> 443 Housing/Accommodations <input type="checkbox"/> 444 Welfare <input type="checkbox"/> 445 Amer. w/Disabilities - Employment <input type="checkbox"/> 446 Amer. w/Disabilities - Other <input type="checkbox"/> 440 Other Civil Rights	PRISONER PETITIONS <input type="checkbox"/> 510 Motions to Vacate Sentence Habeas Corpus: <input type="checkbox"/> 530 General <input type="checkbox"/> 535 Death Penalty <input type="checkbox"/> 540 Mandamus & Other <input type="checkbox"/> 550 Civil Rights <input type="checkbox"/> 555 Prison Condition	OTHER STATUTES <input type="checkbox"/> 400 State Reapportionment <input type="checkbox"/> 410 Antitrust <input type="checkbox"/> 430 Banks and Banking <input type="checkbox"/> 450 Commerce <input type="checkbox"/> 460 Deportation <input type="checkbox"/> 470 Racketeer Influenced and Corrupt Organizations <input type="checkbox"/> 480 Consumer Credit <input type="checkbox"/> 490 Cable/Sat TV <input type="checkbox"/> 810 Selective Service <input type="checkbox"/> 850 Securities/Commodities/Exchange <input type="checkbox"/> 875 Customer Challenge 12 USC 3410 <input type="checkbox"/> 890 Other Statutory Actions <input type="checkbox"/> 891 Agricultural Acts <input type="checkbox"/> 892 Economic Stabilization Act <input type="checkbox"/> 893 Environmental Matters <input type="checkbox"/> 894 Energy Allocation Act <input type="checkbox"/> 895 Freedom of Information Act <input type="checkbox"/> 900 Appeal of Fee Determination Under Equal Access to Justice <input type="checkbox"/> 950 Constitutionality of State Statutes	

V. ORIGIN (Place an "X" in One Box Only)													
<input checked="" type="checkbox"/> 1	Original Proceeding	<input type="checkbox"/> 2	Removed from State Court	<input type="checkbox"/> 3	Remanded from Appellate Court	<input type="checkbox"/> 4	Reinstated or Reopened	<input type="checkbox"/> 5	Transferred from another district (specify)	<input type="checkbox"/> 6	Multidistrict Litigation	<input type="checkbox"/> 7	Appeal to District Judge from Magistrate Judgment

VI. CAUSE OF ACTION	Cite the U.S. Civil Statute under which you are filing (Do not cite jurisdictional statutes unless diversity): <u>35 U.S.C. § 271</u> Brief description of cause: <u>Patent infringement</u>
----------------------------	---

VII. REQUESTED IN COMPLAINT:	<input type="checkbox"/> CHECK IF THIS IS A CLASS ACTION UNDER F.R.C.P. 23	DEMAND \$	CHECK YES only if demanded in complaint: JURY DEMAND: <input checked="" type="checkbox"/> Yes <input type="checkbox"/> No
-------------------------------------	--	------------------	---

VIII. RELATED CASE(S) IF ANY	(See instructions): JUDGE <u>Sleet</u>	DOCKET NUMBER <u>04-876-GMS</u>
-------------------------------------	--	---------------------------------

DATE <u>February 23, 2007</u>	SIGNATURE OF ATTORNEY OF RECORD
----------------------------------	-------------------------------------

FOR OFFICE USE ONLY	RECEIPT # _____	AMOUNT _____	APPLYING IFP _____	JUDGE _____	MAG. JUDGE _____
---------------------	-----------------	--------------	--------------------	-------------	------------------

INSTRUCTIONS FOR ATTORNEYS COMPLETING CIVIL COVER SHEET FORM JS 44**Authority For Civil Cover Sheet**

The JS 44 civil cover sheet and the information contained herein neither replaces nor supplements the filings and service of pleading or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. Consequently, a civil cover sheet is submitted to the Clerk of Court for each civil complaint filed. The attorney filing a case should complete the form as follows:

I. (a) Plaintiffs-Defendants. Enter names (last, first, middle initial) of plaintiff and defendant. If the plaintiff or defendant is a government agency, use only the full name or standard abbreviations. If the plaintiff or defendant is an official within a government agency, identify first the agency and then the official, giving both name and title.

(b) County of Residence. For each civil case filed, except U.S. plaintiff cases, enter the name of the county where the first listed plaintiff resides at the time of filing. In U.S. plaintiff cases, enter the name of the county in which the first listed defendant resides at the time of filing. (NOTE: In land condemnation cases, the county of residence of the "defendant" is the location of the tract of land involved.)

(c) Attorneys. Enter the firm name, address, telephone number, and attorney of record. If there are several attorneys, list them on an attachment, noting in this section "(see attachment)".

II. Jurisdiction. The basis of jurisdiction is set forth under Rule 8(a), F.R.C.P., which requires that jurisdictions be shown in pleadings. Place an "X" in one of the boxes. If there is more than one basis of jurisdiction, precedence is given in the order shown below.

United States plaintiff. (1) Jurisdiction based on 28 U.S.C. 1345 and 1348. Suits by agencies and officers of the United States are included here.

United States defendant. (2) When the plaintiff is suing the United States, its officers or agencies, place an "X" in this box.

Federal question. (3) This refers to suits under 28 U.S.C. 1331, where jurisdiction arises under the Constitution of the United States, an amendment to the Constitution, an act of Congress or a treaty of the United States. In cases where the U.S. is a party, the U.S. plaintiff or defendant code takes precedence, and box 1 or 2 should be marked.

Diversity of citizenship. (4) This refers to suits under 28 U.S.C. 1332, where parties are citizens of different states. When Box 4 is checked, the citizenship of the different parties must be checked. (See Section III below; federal question actions take precedence over diversity cases.)

III. Residence (citizenship) of Principal Parties. This section of the JS 44 is to be completed if diversity of citizenship was indicated above. Mark this section for each principal party.

IV. Nature of Suit. Place an "X" in the appropriate box. If the nature of suit cannot be determined, be sure the cause of action, in Section VI below, is sufficient to enable the deputy clerk or the statistical clerks in the Administrative Office to determine the nature of suit. If the cause fits more than one nature of suit, select the most definitive.

V. Origin. Place an "X" in one of the seven boxes.

Original Proceedings. (1) Cases which originate in the United States district courts.

Removed from State Court. (2) Proceedings initiated in state courts may be removed to the district courts under Title 28 U.S.C., Section 1441. When the petition for removal is granted, check this box.

Remanded from Appellate Court. (3) Check this box for cases remanded to the district court for further action. Use the date of remand as the filing date.

Reinstated or Reopened. (4) Check this box for cases reinstated or reopened in the district court. Use the reopening date as the filing date.

Transferred from Another District. (5) For cases transferred under Title 28 U.S.C. Section 1404(a). Do not use this for within district transfers or multidistrict litigation transfers.

Multidistrict Litigation. (6) Check this box when a multidistrict case is transferred into the district under authority of Title 28 U.S.C. Section 1407. When this box is checked, do not check (5) above.

Appeal to District Judge from Magistrate Judgment. (7) Check this box for an appeal from a magistrate judge's decision.

VI. Cause of Action. Report the civil statute directly related to the cause of action and give a brief description of the cause. **Do not cite jurisdictional statutes unless diversity.** Example: U.S. Civil Statute: 47 USC 553
Brief Description: Unauthorized reception of cable service

VII. Requested in Complaint. Class Action. Place an "X" in this box if you are filing a class action under Rule 23, F.R.Cv.P.

Demand. In this space enter the dollar amount (in thousands of dollars) being demanded or indicate other demand such as a preliminary injunction.

Jury Demand. Check the appropriate box to indicate whether or not a jury is being demanded.

VIII. Related Cases. This section of the JS 44 is used to reference related pending cases if any. If there are related pending cases, insert the docket numbers and the corresponding judge names for such cases.

Date and Attorney Signature. Date and sign the civil cover sheet.

AO FORM 85 RECEIPT (REV. 9/04)

United States District Court for the District of Delaware

Civil Action No. 07 - 113 -

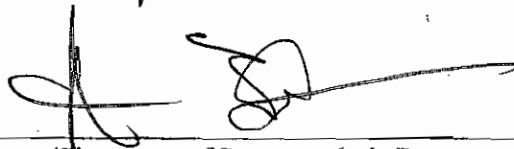
ACKNOWLEDGMENT
OF RECEIPT FOR AO FORM 85

NOTICE OF AVAILABILITY OF A
UNITED STATES MAGISTRATE JUDGE
TO EXERCISE JURISDICTION

I HEREBY ACKNOWLEDGE RECEIPT OF 1 COPIES OF AO FORM 85.

2-23-07

(Date forms issued)



(Signature of Party or their Representative)

Aaron Johnston

(Printed name of Party or their Representative)

Note: Completed receipt will be filed in the Civil Action